iii) o custo de partir de uma entrada vazia “#” e uma saída que corresponde à cadeia de caracteres “r” (`distância[1,0]` que tem um custo 1) mais o custo de remover um “b” da entrada (custo 1), ou seja, um custo total de 2;

Assim, obtém-se a seguinte matriz:

i	4	**a**	4					
	3	**l**	3					
	2	**o**	2					
	1	**b**	1	1				
	0	**#**	0	1	2	3	4	5
			#	**r**	**o**	**l**	**h**	**a**
			0	1	2	3	4	5
			i					

Seguindo o mesmo raciocínio, preenche-se a linha correspondente a i=1:

```
distância[2,1] = Mínimo(1+1 ; 1+1 ; 2+1) = 2
distância[3,1] = Mínimo(2+1 ; 2+1 ; 3+1) = 3
distância[4,1] = Mínimo(3+1 ; 3+1 ; 4+1) = 4
distância[5,1] = Mínimo(4+1 ; 4+1 ; 5+1) = 5
```

i	4	**a**	4					
	3	**l**	3					
	2	**o**	2					
	1	**b**	1	1	2	3	4	5
	0	**#**	0	1	2	3	4	5
			#	**r**	**o**	**l**	**h**	**a**
			0	1	2	3	4	5
			j					

Aplicando o mesmo raciocínio às restantes linhas da matriz, obtém-se:

a	4	4	3	2	2	**2**
l	3	3	2	**1**	**2**	3
o	2	2	**1**	2	3	4
b	1	**1**	2	3	4	5
#	**0**	1	2	3	4	5
	#	**r**	**o**	**l**	**h**	**a**

Escolhendo um caminho válido entre as células inicial e final ([0,0] e [5,4]), seleccionam-se implicitamente as operações que transformam a entrada “bola” na saída “rolha”:

a	4	4	3	2	2	**2**
l	3	3	2	**1**	**2**	3
o	2	2	**1**	2	3	4
b	1	**1**	2	3	4	5
#	**0**	1	2	3	4	5
	#	**r**	**o**	**l**	**h**	**a**

A distância de Levenshtein é 2 (célula [5,4] da matriz) e corresponde à execução das seguintes operações:

i) a substituição de um "b" por um "r" (custo 1);
ii) a substituição de um "o" por um "o" (custo 0);
iii) a substituição de um "l" por um "l" (custo 0);
iv) a inserção de um "h" na saída (custo 1);
v) a substituição de um "a" por um "a" (custo 0);

Problema 10 — Distância mínima entre "intention" e "execution"

O algoritmo usado neste problema é o expresso pelo pseudo-código da função "número_mínimo_edições_2":

```
função número_mínimo_edições_2(saída, entrada)
  n ← comprimento(saída)
  m ← comprimento(entrada)
  cria a matriz distância[n+1, m+1]
  distância[0,0] ← 0
  para cada coluna j de 1 a n faz
    distância[j,0] ← distância[j-1,0] + custo_inserção(saída_j)
  para cada linha i de 1 a m faz
    distância[0,i] ← distância[0,i-1] + custo_remoção(entrada_i)
  para cada linha i de 1 a m faz
    para cada coluna j de 1 a n faz
      distância[j,i] ← min( distância[j-1,i]   + custo_inserção(saída_j),
                            distância[j-1,i-1] + custo_subs(saída_j,entrada_i)
                            distância[j,i-1]   + custo_remoção(entrada_i) )
  devolve distância[n,m]
```

A matriz que resulta da aplicação do algoritmo, quando se considera que cada inserção, remoção e substituição tem custo 1, é a seguinte ("intention" é a palavra de origem e "execution" é a palavra de destino):

n	9	8	8	8	8	8	8	7	6	**5**
o	8	7	7	7	7	7	7	6	**5**	6
i	7	6	6	6	6	6	6	**5**	6	7
t	6	5	5	5	5	5	**5**	6	7	8
n	5	4	4	4	4	**5**	6	7	7	7
e	4	3	4	3	**4**	5	6	6	7	8
t	3	3	3	**3**	4	5	5	6	7	8
n	2	2	**2**	3	4	5	6	7	8	7
i	1	**1**	2	3	4	5	6	6	7	8
#	**0**	1	2	3	4	5	6	7	8	9
	#	**e**	**x**	**e**	**c**	**u**	**t**	**i**	**o**	**n**

Da matriz pode-se concluir que as operações que correspondem ao custo mínimo encontrado são:

i) a substituição de um "i" por um "e" (custo 1);
ii) a substituição de um "n" por um "x" (custo 1);
iii) a substituição de um "t" por um "e" (custo 1);
iv) a substituição de um "e" por um "c" (custo 1);
v) a substituição de um "n" por um "u" (custo 1);
vi) a substituição consecutiva dos caracteres "tion" por "tion" (custo 0);

A matriz que resulta da aplicação do algoritmo, quando se considera que cada inserção e cada remoção tem custo 1 e cada substituição tem custo 2, é a seguinte (“intention” é a palavra de origem e “execution” é a palavra de destino):

n	9	8	9	10	11	12	11	10	9	**8**
o	8	7	8	9	10	11	10	9	**8**	9
i	7	6	7	8	9	10	9	**8**	9	10
t	6	5	6	7	8	9	**8**	9	10	11
n	5	4	5	6	**7**	**8**	9	10	11	10
e	4	3	4	**5**	6	7	8	9	10	9
t	3	4	**5**	6	7	8	7	8	9	8
n	2	3	**4**	5	6	7	8	7	8	7
i	1	**2**	3	4	5	6	7	6	7	8
#	**0**	**1**	2	3	4	5	6	7	8	9
	#	**e**	**x**	**e**	**c**	**u**	**t**	**i**	**o**	**n**

Da matriz pode-se concluir que as operações (uma das alternativas) que correspondem ao custo mínimo encontrado são:

i) a inserção de um “e” (custo 1);
ii) a remoção de um “i” (custo 1);
iii) a substituição de um “n” por um “x” (custo 2);
iv) a remoção de um “t” (custo 1);
v) a substituição de um “e” por um “e” (custo 0);
vi) a substituição de um “n” por “c” (custo 2);
vii) a inserção de um “u” (custo 1);
viii) a substituição consecutiva dos caracteres “tion” por “tion” (custo 0).

Da notar que agora, existem mais caminhos alternativos por uma substituição ter o mesmo custo que:

i) uma inserção seguida de uma remoção;
ii) uma remoção seguida de uma inserção.

Problema 11 — Cálculo de bigramas num corpus com 300 frases

Recordando a definição de bigramas:

$$P(W_n|W_{n-1}) = \frac{C(W_{n-1}W_n)}{\sum_W C(W_{n-1}W)} = \frac{C(W_{n-1}W_n)}{C(W_{n-1})}$$

conclui-se que a tabela fornecida contém os valores a usar no numerador, e os valores a usar no denominador são o número total de ocorrências de cada uma das categorias.

Por exemplo, para calcular o bigrama P(DET|Ø) (ou seja, a probabilidade de ocorrência de um DET, dado que anteriormente ocorreu um início de frase — Ø):

$$P(DET|\phi) = \frac{C(\phi DET)}{C(\phi)} = \frac{213}{300} = 0.710000000000000$$

Os restantes bigramas apresentam-se de seguida:

Par	Contagem do 1º elemento	Contagem do par	Bigrama
Ø,DET	300	213	P(DET|Ø) = 0.710000000000000
Ø,N	300	87	P(N|Ø) = 0.290000000000000
DET,N	558	558	P(N|DET) = 1.000000000000000
N,V	833	358	P(V|N) = 0.429771908763505
N,N	833	108	P(N|N) = 0.129651860744298
N,P	833	366	P(P|N) = 0.439375750300120
V,N	300	75	P(N|V) = 0.250000000000000
V,DET	300	194	P(DET|V) = 0.646666666666667
P,DET	307	226	P(DET|P) = 0.736156351791531
P,N	307	81	P(N|P) = 0.263843648208469

Problema 12 — Algoritmo de Viterbi em "O rato roeu a rolha"

[Este problema é uma adaptação de um trabalho realizado pelo aluno Sérgio Costa em 2005/06]

O algoritmo de Viterbi, criado por Andrew Viterbi, é um algoritmo de programação dinâmica que permite encontrar a sequência mais provável de estados não observáveis – o chamado "caminho Viterbi" – resultando numa sequência de eventos observáveis, utilizada normalmente no contexto dos modelos de Markov não observáveis.

Actualmente, o algoritmo de Viterbi é frequentemente usado em áreas como o reconhecimento de discurso, detecção de palavras-chave, linguística computacional e bio-informática. Neste exercício, pretende-se estudar a aplicação do algoritmo de Viterbi na resolução do problema da etiquetação morfológica.

O algoritmo não é geral, na medida em que faz uma série de assumpções. Em primeiro lugar, tanto os eventos observáveis como os não observáveis têm de estar em sequência, sendo que estas sequências correspondem, normalmente, a sequências temporais. Em segundo lugar, as duas sequências devem estar alinhadas, e um evento observável tem de corresponder, obrigatoriamente, a um evento não observável. Em terceiro lugar, o cálculo da sequência não observável mais provável num determinado ponto `t`, deve depender apenas do evento observável no ponto `t` e da sequência mais provável no ponto `t−1`. Todas estas assumpções são satisfeitas por um modelo de Markov de primeira ordem.

O algoritmo opera com base em máquinas de estados, existindo um número finito de estados, que pode ser enumerado. Cada estado (ou nó) pode ser alcançado por uma ou mais sequências (ou caminhos). Contudo, existe uma sequência mais provável, o chamado "caminho sobrevivente" (survivor path), que permite alcançar esse estado. Esta é uma das assumpções fundamentais do algoritmo, pois este analisa todos os caminhos que levam a um estado, e escolhe aquele que tem maior probabilidade. Desta forma, o algoritmo não tem de manter o registo de todos os caminhos alternativos, armazenando apenas um caminho por estado. A segunda assumpção chave do algoritmo é o facto de a transição de um estado para um novo estado ser marcada por uma métrica incremental, usualmente um número. Outra das assumpções chaves do algoritmo é o facto de os eventos serem cumulativos ao longo de um caminho, sendo, normalmente, aditivos. Deste modo, o ponto fulcral do algoritmo consiste em associar um número a cada estado. Quando ocorre um evento, o algoritmo passa para um novo conjunto de estados, combinando a métrica de um estado possível anterior com a métrica incremental da transição resultante da ocorrência do evento. Neste passo, o algoritmo escolhe a melhor transição. Note-se que a métrica incremental, associada a um evento, depende da possibilidade de se realizar a transição do estado anterior para o novo estado.

Durante a aplicação do algoritmo, é necessário armazenar um historial da procura. Em alguns casos, o historial da procura é finito, mas pode dar-se o caso de não ser possível armazenar todo o historial e, deste modo, ser necessário limitar a profundidade da procura.

O algoritmo de Viterbi recorre a três estruturas de dados (todas elas vectores):

- `SeqScore` – regista a "pontuação" da melhor sequência encontrada até à posição `t`, com a categoria L_n. O vector tem dimensões [`N` x `n`], em que `N` é o número de categorias léxicas e `n` é o número de palavras na sequência;
- `BackPtr` – regista o estado anterior a um dado estado. O vector tem dimensões [`N` x `n`];
- `C` – regista o resultado, isto é, a melhor sequência de etiquetas. O vector tem dimensões [`1` x `n`].

O algoritmo pode ser divido em três etapas: Iniciação, Iteração e Identificação da Sequência. Assim, dada a sequência de palavras $w_1,\dots,w_n$, as categorias léxicas $L_1,\dots,L_N$, encontra-se a sequência mais provável de categorias léxicas $C = [C_1,\dots,C_n]$ do seguinte modo:

```
Iniciação
Para i=1 até N faz
    SeqScore[i,1] = P(w1|Li)*P(Li|Ø)
    BackPtr[i,1] = 0
Iteração
Para t=2 até n faz
    Para i=1 até N faz
        SeqScore[i,t] = MAXj=1,N(SeqScore[j,t-1]*P(Li|Lj))*P(wt|Li)
        BackPtr[i,t] = índice de j que resultou na pontuação máxima
Identificação da Sequência
    C[n] = I que maximiza SeqScore[i,n]
    Para i=n-1 até 1 faz
        C[i] = BackPtr[C(i+1),i+1]
```

No pseudo-código representado acima:

- Ø representa uma categoria léxica fictícia, utilizada apenas para o início da frase;
- Na etapa de iniciação, a primeira coluna do vector BackPtr toma o valor zero em todas as linhas, pois nenhuma palavra precede a primeira palavra;
- Na etapa final, de identificação da sequência de etiquetas, encontra-se, na última coluna do vector SeqScore, a entrada com o valor mais alto. O índice dessa entrada indica a categoria léxica à qual pertence a última palavra;
- As categorias léxicas das palavras anteriores são encontradas fazendo retrocesso com o vector BackPtr, isto é, a entrada do vector o vector BackPtr[C(i+1),i+1] indica a categoria léxica da palavra i.

As categorias léxicas têm a seguinte codificação: L_1=N, L_2=V, L_3=DET, L_4=P.

A execução do algoritmo começa na etapa de iniciação, pelo que é necessário calcular as probabilidades de geração léxica (lexical-generation probabilities) que são estimadas através da contagem do número de ocorrências de cada palavra, por categoria (dados fornecidos no enunciado):

P(o|DET)	250/558 = 4.48028673835E-01
P(o|P)	87/307 = 2.83387622150E-01
P(rato|N)	28/833 = 3.36134453782E-02
P(roeu|V)	1/300 = 3.33333333333E-03
P(a|DET)	208/558 = 3.72759856631E-01
P(a|P)	54/307 = 1.75895765472E-01
P(rolha|N)	5/833 = 6.00240096038E-03

P(DET|Ø)	0.710000000000000
P(N|Ø)	0.290000000000000
P(N|DET)	1.000000000000000
P(V|N)	0.429771908763505
P(N|N)	0.129651860744298
P(P|N)	0.439375750300120
P(N|V)	0.250000000000000
P(DET|V)	0.646666666666667
P(DET|P)	0.736156351791531
P(N|P)	0.263843648208469

A tabela mais à direita resume os resultados referentes ao corpus de 300 frases. Assume-se que qualquer bigrama não listado tem probabilidade 1.0E-06.

Agora já se pode preencher a primeira coluna dos vectores SeqScore e BackPtr:

```
SeqScore[1,1]=P(o|N)*P(N|Ø)= 0.000001*0.29 = 2.9E-07
BackPtr[1,1]=0
SeqScore[2,1]=P(o|V)*P(V|Ø)= 0.000001*0.000001 = 1.0E-12
BackPtr[2,1]=0
SeqScore[3,1]=P(o|DET)*P(DET|Ø)= 0.425170068027*0.71= 3.18100358423E-01
BackPtr[3,1]=0
SeqScore[4,1]=P(o|P)*P(P|Ø)=0.28338762214983*0.000001= 2.83387622150E-07
BackPtr[4,1]=0
```

Apresenta-se em seguida, na forma de uma tabela, o cálculo realizado na etapa de iteração:

t	i	SeqScore[i,t]	BackPtr[i,t]
2	1	MAX(SeqScore[1,1]*P(N|N), SeqScore[2,1]*P(N|V), SeqScore[3,1]*P(N|DET), SeqScore[4,1]*P(N|P)) *P(rato|N) = 1.06924490226E-02	3
2	2	MAX(SeqScore[1,1]*P(V|N), SeqScore[2,1]*P(V|V), SeqScore[3,1]*P(V|DET), SeqScore[4,1]*P(V|P)) *P(rato|V) = 3.18100358423E-13	3
2	3	MAX(SeqScore[1,1]*P(DET|N), SeqScore[2,1]*P(DET|V), SeqScore[3,1]*P(DET|DET),SeqScore[4,1]*P(DET|P)) *P(rato|DET) = 3.18100358423E-13	3
2	4	MAX(SeqScore[1,1]*P(P|N), SeqScore[2,1]*P(P|V), SeqScore[3,1]*P(P|DET), SeqScore[4,1]*P(P|P)) *P(rato|P) = 3.18100358423E-13	3
3	1	MAX(SeqScore[1,2]*P(N|N), SeqScore[2,2]*P(N|V), SeqScore[3,2]*P(N|DET), SeqScore[4,2]*P(N|P)) *P(roeu|N) = 1.38629591170E-09	1
3	2	MAX(SeqScore[1,2]*P(V|N), SeqScore[2,2]*P(V|V), SeqScore[3,2]*P(V|DET), SeqScore[4,2]*P(V|P)) *P(roeu|V) = 1.53177140860E-05	1
3	3	MAX(SeqScore[1,2]*P(DET|N), SeqScore[2,2]*P(DET|V), SeqScore[3,2]*P(DET|DET), SeqScore[4,2]*P(DET|P)) *P(roeu|DET) = 1.06924490226E-14	1
3	4	MAX(SeqScore[1,2]*P(P|N), SeqScore[2,2]*P(P|V), SeqScore[3,2]*P(P|DET), SeqScore[4,2]*P(P|P)) *P(roeu|P) = 4.69800281186E-09	1
4	1	MAX(SeqScore[1,3]*P(N|N), SeqScore[2,3]*P(N|V), SeqScore[3,3]*P(N|DET), SeqScore[4,3]*P(N|P)) *P(a|N) = 3.82942852151E-12	2
4	2	MAX(SeqScore[1,3]*P(V|N), SeqScore[2,3]*P(V|V), SeqScore[3,3]*P(V|DET), SeqScore[4,3]*P(V|P)) *P(a|V) = 5.95791040081E-16	1
4	3	MAX(SeqScore[1,3]*P(DET|N), SeqScore[2,3]*P(DET|V), SeqScore[3,3]*P(DET|DET), SeqScore[4,3]*P(DET|P)) *P(a|DET) = 3.69235602628E-06	2
4	4	MAX(SeqScore[1,3]*P(P|N), SeqScore[2,3]*P(P|V), SeqScore[3,3]*P(P|DET), SeqScore[4,3]*P(P|P)) *P(a|P) = 1.07138956164E-10	1
5	1	MAX(SeqScore[1,4]*P(N|N), SeqScore[2,4]*P(N|V), SeqScore[3,4]*P(N|DET), SeqScore[4,4]*P(N|P)) *P(rolha|N) = 2.21630013582E-08	3
5	2	MAX(SeqScore[1,4]*P(V|N), SeqScore[2,4]*P(V|V), SeqScore[3,4]*P(V|DET), SeqScore[4,4]*P(V|P)) *P(rolha|V) = 3.69235602628E-18	3
5	3	MAX(SeqScore[1,4]*P(DET|N), SeqScore[2,4]*P(DET|V), SeqScore[3,4]*P(DET|DET), SeqScore[4,4]*P(DET|P)) *P(rolha|DET) = 7.88710231044E-17	4
5	4	MAX(SeqScore[1,4]*P(P|N), SeqScore[2,4]*P(P|V), SeqScore[3,4]*P(P|DET), SeqScore[4,4]*P(P|P)) *P(rolha|P) = 3.69235602628E-18	3

Deste modo, o conteúdo final dos vectores SeqScore e BackPtr são, respectivamente:

SeqScore		n (palavras)				
		1 (O)	3 (rato)	3 (roeu)	4 (a)	5 (rolha)
N (categorias)	1 (N)	2.9E-07	1.1E-02	1.4E-09	3.9E-12	2.2E-08
	2 (V)	1.0E-12	3.2E-13	1.5E-05	6.0E-16	3.7E-18
	3 (DET)	3.2E-01	3.2E-13	1.1E-14	3.7E-06	7.9E-17
	4 (P)	2.8E-07	3.2E-13	4.7E-09	1.1E-10	3.7E-18

BackPtr		n (palavras)				
		1 (O)	2 (rato)	3 (roeu)	4 (a)	5 (rolha)
N (categorias)	1 (N)	0	3	1	2	3
	2 (V)	0	3	1	1	3
	3 (DET)	0	3	1	2	4
	4 (P)	0	3	1	1	3

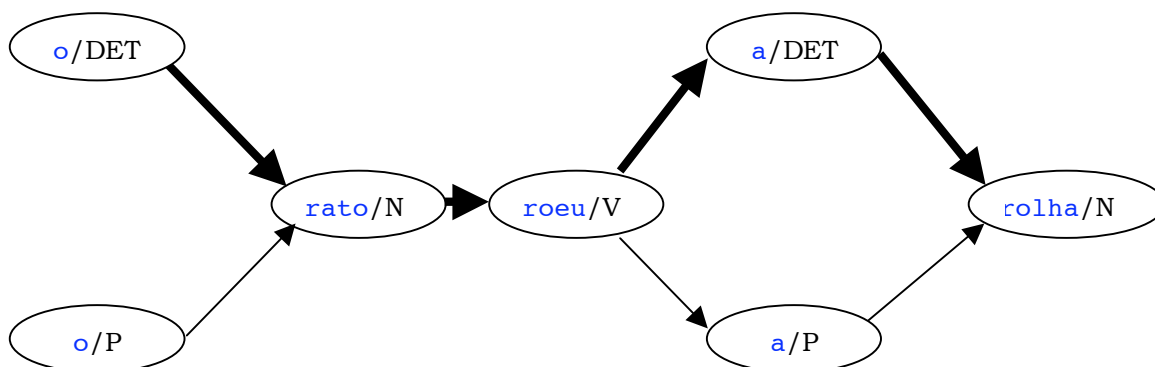
Terminada a etapa de iteração, falta apenas construir a sequência de etiquetas mais provável. Começamos por notar que o valor mais alto para a última coluna de `SeqScore` encontra-se na primeira linha. Deste modo, `C(n)=C(5)=1`, isto é, a etiqueta da última palavra (`rolha`) é `N` (nome). As restantes etiquetas calculam-se do seguinte modo:

```
C[4] = backPtr[C[5],5] = backPtr[1,5] = 3
C[3] = backPtr[C[4],4] = backPtr[3,4] = 2
C[2] = backPtr[C[3],3] = backPtr[2,3] = 1
C[1] = backPtr[C[2],2] = backPtr[1,2] = 3
```

Ou seja, o conteúdo final do vector `C` é:

C	n (palavras)				
	1 (O)	2 (rato)	3 (roeu)	4 (a)	5 (rolha)
	3	1	2	3	1

Assim `C=[3,1,2,3,1]` o que equivale a dizer que `o` foi etiquetado como `DET` (determinante), `rato` foi etiquetado como `N` (nome), `roeu` foi etiquetado como `V` (verbo), `a` foi etiquetado como `DET` (determinante) e `rolha` foi etiquetado como `N` (nome).

Em termos gráficos, o resultado da *etapa de iteração* é o seguinte (as setas mais carregadas representam a sequência mais provável, o resultado da *identificação da sequência*).

Problema 13 — Análise descendente de "Os alunos dedicados passam"

$_1$ Os $_2$ alunos $_3$ dedicados $_4$ passam $_5$

	Estado Corrente	Estados Alternativos
1	**((F) 1)**	**()**
2	**((SN SV) 1)**	()
3	**((Det N SV) 1)**	**((Det N Adj SV) 1)**
4	**((N SV) 2)**	((Det N Adj SV) 1)
5	**((SV) 3)**	((Det N Adj SV) 1)
6	**((V) 3)**	**((V SN) 3)** ((Det N Adj SV) 1)
7	**((V SN) 3)**	((Det N Adj SV) 1)
8	**((Det N Adj SV) 1)**	()
9	**((N Adj SV) 2)**	()
10	**((Adj SV) 3)**	()
11	**((SV) 4)**	()
12	**((V) 4)**	**((V SN) 4)**
13	**(() 5)**	((V SN) 4)

Sucesso!

Problema 14 — Análise descendente de "*Os professores dão as aulas em as salas*"

$_1$ Os $_2$ professores $_3$ dão $_4$ as $_5$ aulas $_6$ em $_7$ as $_8$ salas $_9$

	Estado Corrente	Estados Alternativos
1	((F) 1)	()
2	((SN SV) 1)	*…sem alteração.*
3	((SNA SV) 1)	((SNA SP SV) 1)
4	((Det N SV) 1)	((Det N Adj SV) 1) ((Det NP SV) 1) ((SNA SP SV) 1)
5	((N SV) 2)	*…sem alteração.*
6	((SV) 3)	*…sem alteração.*
7	((V) 3)	((V SN) 3) ((Det N Adj SV) 1) ((Det NP SV) 1) ((SNA SP SV) 1)
8	(() 4)	*…sem alteração.*
9	((V SN) 3)	((Det N Adj SV) 1) ((Det NP SV) 1) ((SNA SP SV) 1)
10	((SN) 4)	*…sem alteração.*
11	((SNA) 4)	((SNA SP) 4) ((Det N Adj SV) 1) ((Det NP SV) 1) ((SNA SP SV) 1)
12	((Det N) 4)	((Det N Adj) 4) ((Det NP) 4) ((SNA SP) 4) ((Det N Adj SV) 1) ((Det NP SV) 1) ((SNA SP SV) 1)
13	((N) 5)	*…sem alteração.*
14	(() 6)	*…sem alteração.*
15	((Det N Adj) 4)	((Det NP) 4) ((SNA SP) 4) ((Det N Adj SV) 1) ((Det NP SV) 1) ((SNA SP SV) 1)
16	((N Adj) 5)	*…sem alteração.*
17	((Adj) 6)	*…sem alteração.*
18	((Det NP) 4)	((SNA SP) 4) ((Det N Adj SV) 1) ((Det NP SV) 1) ((SNA SP SV) 1)
19	((NP) 5)	*…sem alteração.*
20	((SNA SP) 4)	((Det N Adj SV) 1) ((Det NP SV) 1) ((SNA SP SV) 1)
21	((Det N SP) 4)	((Det N Adj SP) 4) ((Det NP SP) 4) ((Det N Adj SV) 1) ((Det NP SV) 1) ((SNA SP SV) 1)
22	((N SP) 5)	*…sem alteração.*
23	((SP) 6)	*…sem alteração.*
24	((P SN) 6)	*…sem alteração.*
25	((SN) 7)	*…sem alteração.*
26	((SNA) 7)	((SNA SP) 7) ((Det N Adj SP) 4) ((Det NP SP) 4) ((Det N Adj SV) 1) ((Det NP SV) 1) ((SNA SP SV) 1)
27	((Det N) 7)	((Det N Adj) 7) ((Det NP) 7) ((SNA SP) 7) ((Det NP SP) 4) ((Det N Adj SP) 4) ((Det N Adj SV) 1) ((Det NP SV) 1) ((SNA SP SV) 1)
28	((N) 8)	*…sem alteração.*
29	(() 9)	*…sem alteração.*

Sucesso!

Problema 15 — Análise ascendente (grafo) de "*A aluna de saia entrou pelas traseiras*"

F_6 (SN_4,SV_5)

F_5 (SN_5,SV_5)

F_4 (SN_3,SV_5)

F_3 (SN_4,SV_4)

F_2 (SN_5,SV_4)

SN_5 (N_1,SP_1)

SN_4 (Det_1,SN_1,SP_1)

F_1 (SN_3,SV_4)

SV_2 (V_1) | SV_3 (V_2) | SV_5 (SV_4,SP_2)

SV_1 (N_1,SP_1)

SN_2 (N_1) | SP_1 (P_1,SN_3)

SN_1 (Det_1,N_1) | SN_3 (N_2) | SP_2 (P_2,SN_6)

V_1 | V_2 | SV_4 | SN_6 (N_3)

Det_1 | N_1 | P_1 | N_2 | V_3 | P_2 | N_3

1 a 2 aluna 3 de 4 saia 5 entrou 6 pelas 7 traseiras 8

SN → Det_1 • N

SN → Det_1 • N SP

SN → Det_1 N_1 • SP

F → SN_1 • SV

F → SN_2 • SV

SN → N_1 • SP

SV → V_1 • SP

SP → P_1 • SN

F → SN_4 • SV

F → SN_5 • SV

F → SN_3 • SV

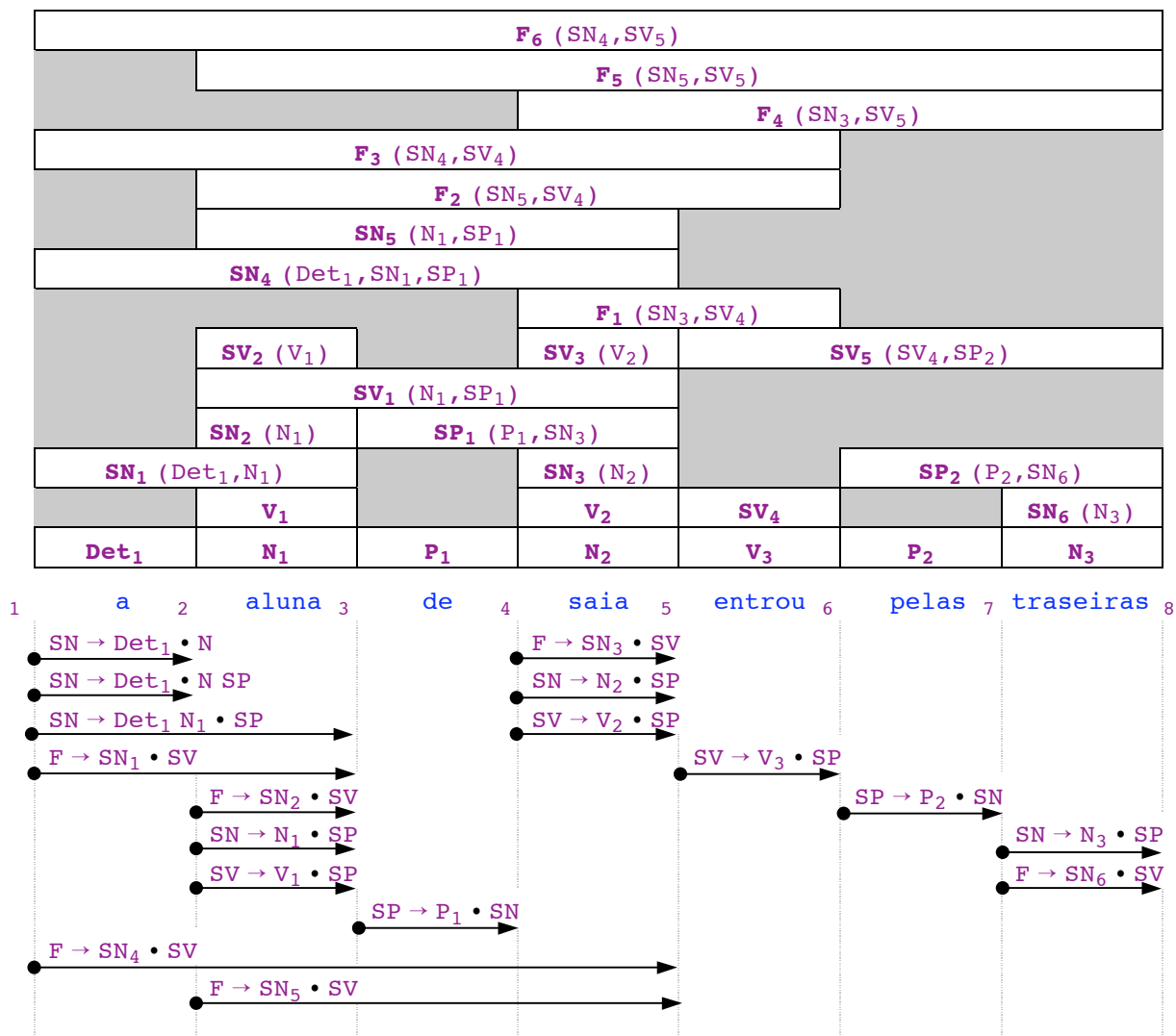
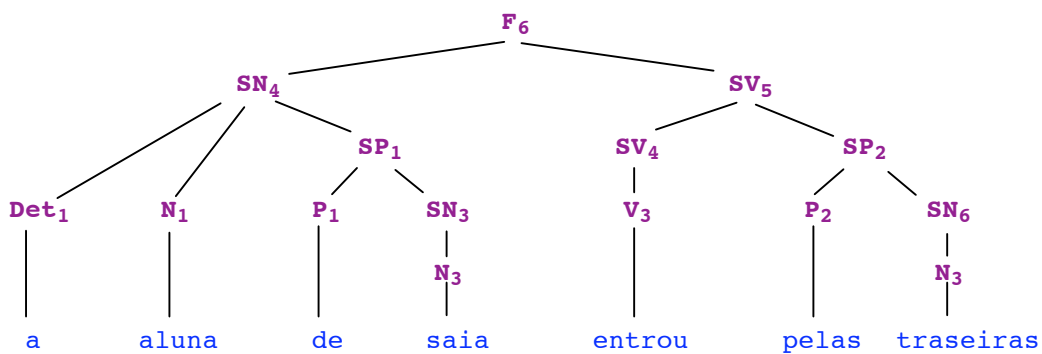
SN → N_2 • SP

SV → V_2 • SP

SV → V_3 • SP

SP → P_2 • SN

SN → N_3 • SP

F → SN_6 • SV

Estruturas sintáctica correspondente:

Problema 16 — Análise ascendente (grafo) de "*Senhora aluga quarto a rapariga em bom estado*"

Estruturas sintácticas correspondentes:

Algumas considerações sobre as interpretações que se podem associar a cada uma destas estruturas:

F_5: **Aluguer:** em bom estado (o quarto!)
Quarto: para rapariga

F_6: **Aluguer:** a uma rapariga
Quarto: em bom estado

F_7: **Aluguer:** um quarto
Quarto: para rapariga em bom estado

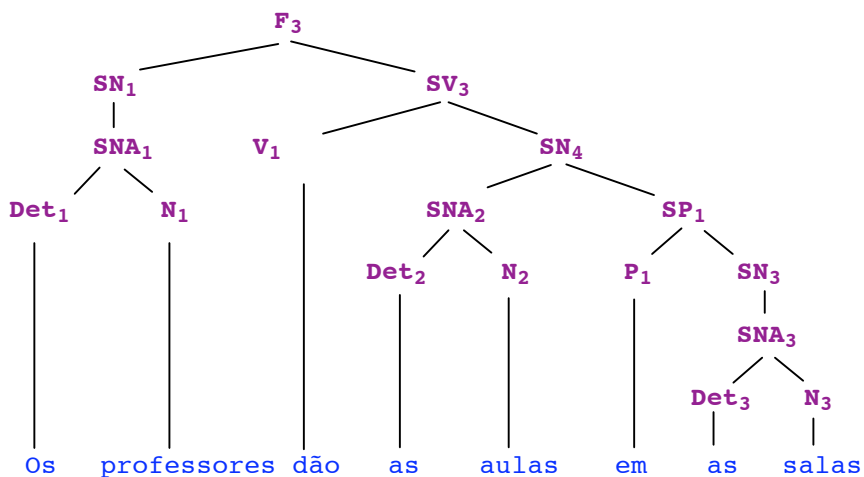
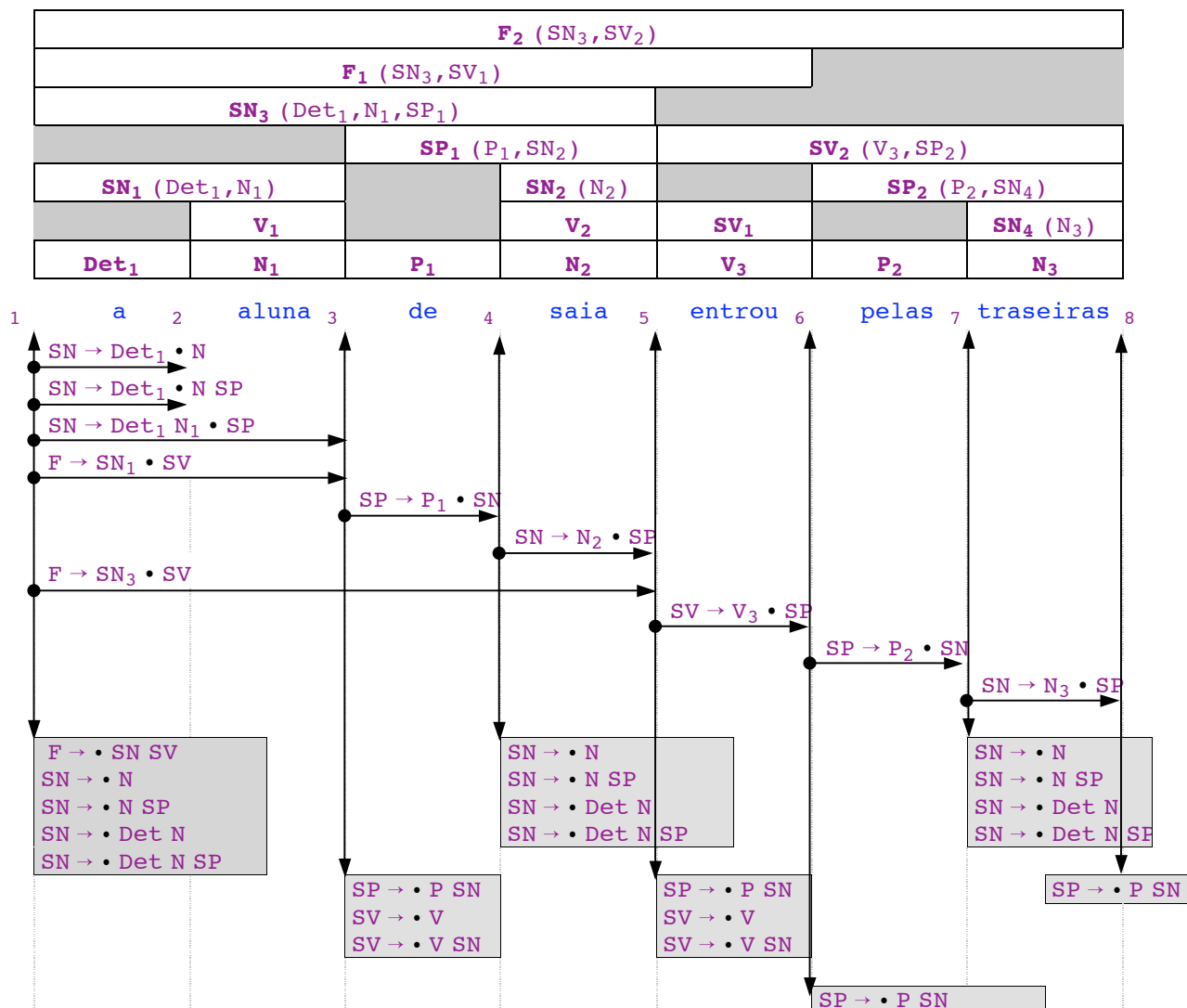
F_8: **Aluguer:** um quarto a uma rapariga
Quarto: ?
Rapariga: em bom estado

Problema 17 — Análise ascendente (grafo) de "*Os professores dão as aulas em as salas*"

<table>
<tr><td colspan="8">$\mathbf{F_3}$ (SN_1,SV_3)</td></tr>
<tr><td colspan="2"></td><td colspan="6">$\mathbf{SV_3}$ (V_1,SN_4)</td></tr>
<tr><td colspan="3"></td><td colspan="5">$\mathbf{SN_4}$ (SNA_2,SP_1)</td></tr>
<tr><td colspan="5">$\mathbf{F_2}$ (SN_1,SV_2)</td><td colspan="3" rowspan="2"></td></tr>
<tr><td colspan="2"></td><td colspan="3">$\mathbf{SV_2}$ (V_1,SN_2)</td></tr>
<tr><td colspan="3">$\mathbf{F_1}$ (SN_1,SV_1)</td><td colspan="2"></td><td colspan="3">$\mathbf{SP_1}$ (P_1,SN_3)</td></tr>
<tr><td colspan="2">$\mathbf{SN_1}$ (SNA_1)</td><td rowspan="2"></td><td colspan="2">$\mathbf{SN_2}$ (SNA_2)</td><td rowspan="2"></td><td colspan="2">$\mathbf{SN_3}$ (SNA_3)</td></tr>
<tr><td colspan="2">$\mathbf{SNA_1}$ (Det_1,N_1)</td><td colspan="2">$\mathbf{SNA_2}$ (Det_2,N_2)</td><td colspan="2">$\mathbf{SNA_3}$ (Det_3,N_3)</td></tr>
<tr><td>$\mathbf{Pron_1}$</td><td></td><td>$\mathbf{SV_1}$ (V_1)</td><td>$\mathbf{Pron_2}$</td><td colspan="2"></td><td>$\mathbf{Pron_3}$</td><td></td></tr>
<tr><td>$\mathbf{Det_1}$</td><td>$\mathbf{N_1}$</td><td>$\mathbf{V_1}$</td><td>$\mathbf{Det_2}$</td><td>$\mathbf{N_2}$</td><td>$\mathbf{P_1}$</td><td>$\mathbf{Det_3}$</td><td>$\mathbf{N_3}$</td></tr>
</table>

1 os 2 profess. 3 dão 4 as 5 aulas 6 em 7 as 8 salas 9

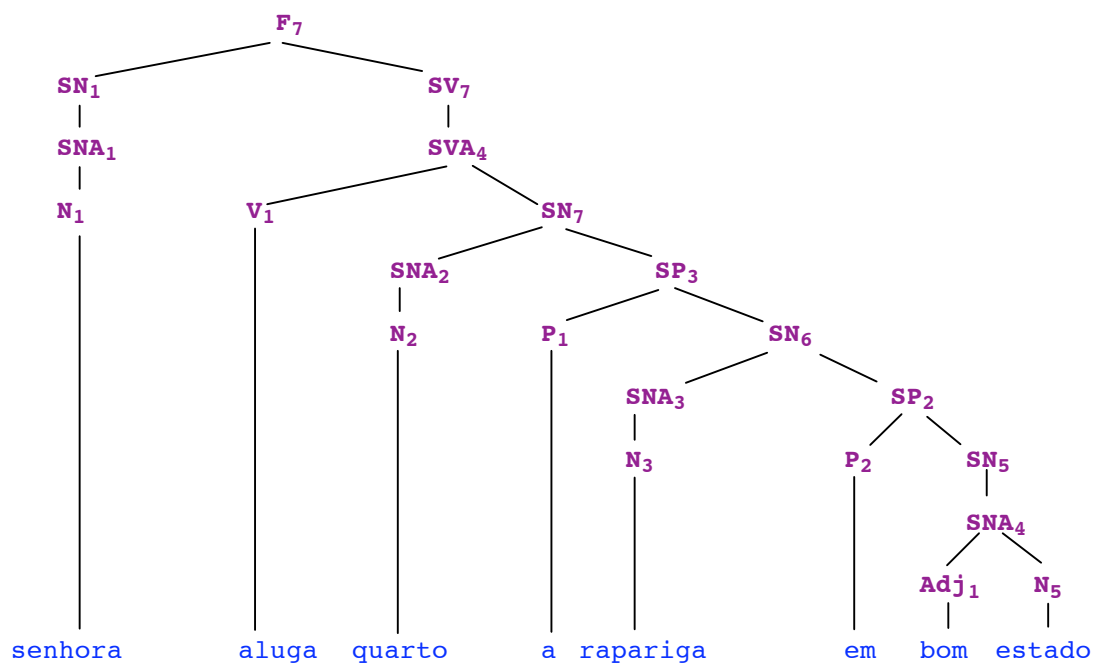
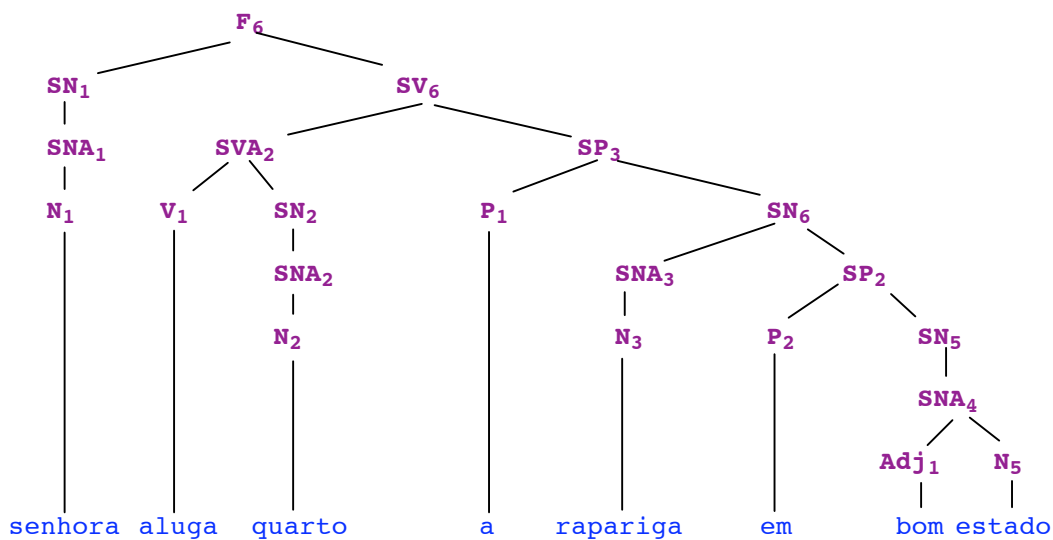
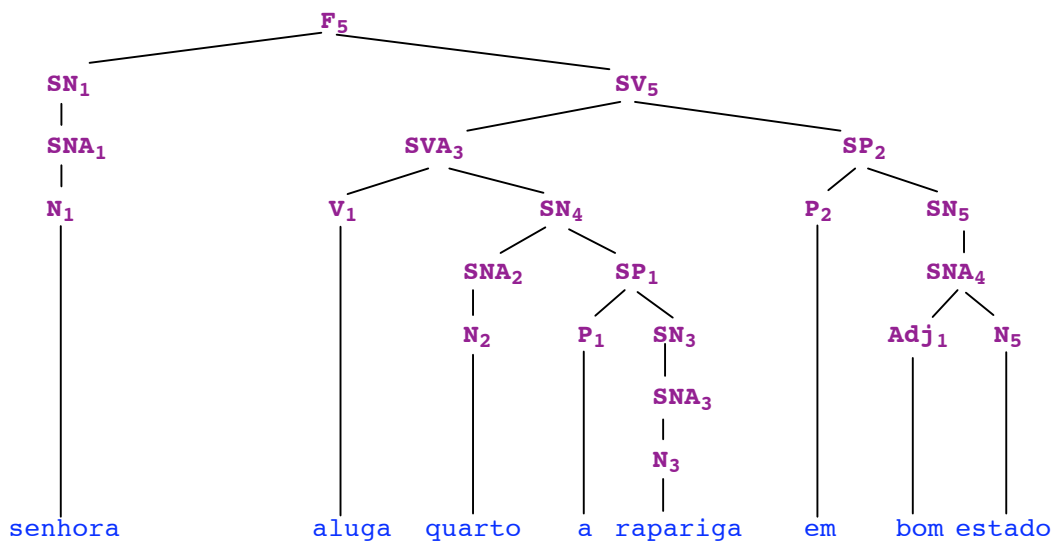
- (1→2) SNA → Det_1 • N
- (1→2) SNA → Det_1 • N Adj
- (1→2) SNA → Det_1 • NP
- (1→3) SNA → Det_1 N_1 • Adj
- (1→3) SN → SNA_1 • SP
- (1→3) F → SN_1 • SV
- (3→4) SV → V_1 • SN
- (4→5) SNA → Det_2 • N
- (4→5) SNA → Det_2 • N Adj
- (4→5) SNA → Det_2 • NP
- (4→6) SNA → Det_2 N_2 • Adj
- (4→6) SN → SNA_2 • SP
- (4→6) F → SN_2 • SV
- (6→7) SP → P_1 • SN
- (4→9) F → SN_4 • SV
- (7→8) SNA → Det_3 • N
- (7→8) SNA → Det_3 • N Adj
- (7→8) SNA → Det_3 • NP
- (7→9) SNA → Det_3 N_3 • Adj
- (7→9) SN → SNA_3 • SP
- (7→9) F → SN_3 • SV

A que corresponde a seguinte estrutura sintáctica:

Problema 18 — Análise descendente (grafo) de "*A aluna de saia entrou pelas traseiras*"

A que corresponde a estrutura sintáctica representada na página 40.

Problema 19 — Análise descendente (grafo) de "*Senhora aluga quarto a rapariga em bom estado*"

As estruturas sintácticas correspondentes não são coincidentes com as que encontradas no Problema 16 porque a gramática agora utilizada é levemente diferente é mais restritiva em relação aos sintagmas verbais (SV):

Problema 20 — Análise descendente (grafo) de "*Os professores dão as aulas em as salas*"

A que corresponde a estrutura sintáctica representada na página 44.

Problema 21 — Análise com algoritmo de Earley de "A aluna de saia entrou pelas traseiras"

O Algoritmo de Earley recorre a três funções:

- Predictor: adiciona predições ao grafo. É activado quando o ponto "•" está à frente de um símbolo não terminal, ou seja um símbolo que não é uma etiqueta morfológica;
- Completer: move o ponto uma posição para a direita sempre que são encontrados novos constituintes. É activado quando o ponto está no fim de um estado;
- Scanner: lê as palavras de entrada e introduz estados representando essas palavras no grafo. É activado quando o ponto (num estado) está à frente de um símbolo que é uma etiqueta morfológica.

Para ajudar a interpretação da aplicação do algoritmo, usa-se uma seta vermelha ("→") para representar estados introduzidos pela função Predictor, uma seta verde ("→") para representar estados introduzidos pela função Completer e uma seta laranja ("→") para representar estados introduzidos pela função Scanner.

Apresenta-se de seguida o algoritmo de Earley:

```
function EARLEY-PARSE(words,grammar) returns chart
    ENQUEUE((δ→•S, [0,0], chart[0])
    for i from 0 to LENGTH(words) do
        for each state in chart[i] do
            if INCOMPLETE?(state) and NEXT-CAT(state) is not a POS then
                PREDICTOR(state)
            elseif INCOMPLETE?(state) and NEXT-CAT(state) is a POS then
                SCANNER(state)
            else
                COMPLETER(state)
        end
    end
    return(chart)

procedure PREDICTOR((A→α•Bσ, [i,j]))
    for each (B→γ) in GRAMMAR-RULES-FOR(B,grammar) do
        ENQUEUE((B→•γ, [j,j]), chart[j])
    end
end

procedure SCANNER((A→α•Bσ, [i,j]))
    if (B ∈ PARTS-OF-SPEECH(word[j]) then
        ENQUEUE((B→word[j]• , [j,j+1]), chart[j+1])
    end
end

procedure COMPLETER((B→γ•, [j,k]))
    for each (A→α•Bσ, [i,j]) in chart[j] do
        ENQUEUE((A→αB•σ, [i,k]), chart[k])
    end
end

procedure ENQUEUE(state,chart-entry)
    if state is not already in chart-entry then
        Add state at the end of chart-entry
    end
end
```

Chart[0]	
δ →•F	[0,0]
F →•SN SV	[0,0]
SN→•N	[0,0]
SN→•N SP	[0,0]
SN→•Det N	[0,0]
SN→•Det N SP	[0,0]

Chart[1]	
Det→a•	[0,1]
SN→Det•N	[0,1]
SN→Det•N SP	[0,1]

Chart[2]	
N→aluna•	[1,2]
SN→Det N•	[0,2]
SN→Det N•SP	[0,2]
SP→•P SN	[2,2]

Chart[3]	
P→de•	[2,3]
SP→P•SN	[2,3]
SN→•N	[3,3]
SN→•N SP	[3,3]
SN→•Det N	[3,3]
SN→•Det N SP	[3,3]

Chart[4]	
N→saia•	[3,4]
SN→N•	[3,4]
SN→N•SP	[3,4]
SP→P SN•	[2,4]
SN→Det N SP•	[0,4]
F →SN•SV	[0,4]
SV→•V	[4,4]
SV→•V SP	[4,4]

Chart[5]	
V→entrou•	[4,5]
SV→V•	[4,5]
SV→V•SP	[4,5]
F →SN SV•	[0,5]
δ →F•	[0,5]
SP→•P SN	[5,5]

Chart[6]	
P→pelas•	[5,6]
SP→P•SN	[5,6]
SN→•N	[6,6]
SN→•N SP	[6,6]
SN→•Det N	[6,6]
SN→•Det N SP	[6,6]

Chart[7]	
N→traseiras•	[6,7]
SN→N•	[6,7]
SN→N•SP	[0,1]
SP→P SN•	[5,7]
SV→V SP•	[4,7]
F →SN SV•	[0,7]
δ →F•	[0,7]

Na última linha derivou-se o símbolo inicial (“δ”) usando todas as palavras da frase (“[0,7]”), a que corresponde a estrutura sintáctica representada na página 40.

Problema 22 — Análise com algoritmo de Earley "Senhora aluga quarto a rapariga em bom estado"

Chart[0]	
δ →•F	[0,0]
F →•SN SV	[0,0]
SN→•SNA	[0,0]
SN→•SNA SP	[0,0]
SNA→•Adj N	[0,0]
SNA→•N	[0,0]

Chart[1]	
N→Senhora•	[0,1]
SNA→N•	[0,1]
SN→SNA•	[0,1]
SN→SNA•SP	[0,1]
F→SN•SV	[0,1]
SP→•P SN	[1,1]
SV→•SVA SN	[1,1]
SV→•SVA SP	[1,1]
SVA→•V	[1,1]
SVA→•V SN	[1,1]

Chart[2]	
V→aluga•	[1,2]
SVA→V•	[1,2]
SVA→V•SN	[1,2]
SN→•SNA	[2,2]
SN→•SNA SP	[2,2]
SNA→•Adj N	[2,2]
SNA→•N	[2,2]

Chart[3]	
N→quarto•	[2,3]
SNA→N•	[2,3]
SN→SNA•	[2,3]
SN→SNA•SP	[2,3]
SVA→V SN•	[1,3]
SV→SVA•SN	[1,3]
SV→SVA•SP	[1,3]
SN→•SNA	[3,3]
SN→•SNA SP	[3,3]
SNA→•Adj N	[3,3]
SNA→•N	[3,3]
SP→•P SN	[3,3]

Chart[4]	
P→a•	[3,4]
SP→P•SN	[3,4]
SN→•SNA	[4,4]
SN→•SNA SP	[4,4]
SNA→•Adj N	[4,4]
SNA→•N	[4,4]

Chart[5]	
N→rapariga•	[4,5]
SNA→N•	[4,5]
SN→SNA•	[4,5]
SN→SNA•SP	[4,5]
SP→P SN•	[3,5]
SN→SNA SP•	[2,5]
SV→SVA SP•	[1,5]
SVA→V SN•	[1,5]
F→SN SV•	[0,5]
δ→F•	[0,5]
SP→•P SN	[5,5]

Chart[6]	
P→em•	[5,6]
SP→P•SN	[5,6]
SN→•SNA	[6,6]
SN→•SNA SP	[6,6]
SNA→•Adj N	[6,6]
SNA→•N	[6,6]

Chart[7]	
Adj→bom•	[6,7]
SNA→Adj•N	[6,7]

Chart[8]	
N→estado•	[7,8]
SNA→Adj N•	[6,8]
SN→SNA•	[6,8]
SN→SNA•SP	[6,8]
SP→P SN•	[5,8]
SN→SNA SP•	[4,8]
SP→P SN•	[3,8]
SN→SNA SP•	[2,8]
SV→SVA SP•	[1,8]
SVA→V SN•	[1,8]
F→SN SV•	[0,8]
SV→SVA•SN	[1,8]
SV→SVA•SP	[1,8]
δ →F•	[0,8]

Na última linha derivou-se o símbolo inicial ("δ") usando todas as palavras da frase ("[0,8]"), a que corresponde a estrutura sintáctica representada na página 46.

Problema 23 — Análise com algoritmo de Earley "Os professores dão as aulas em as salas"

Para obter a árvore de análise é necessário incrementar a representação de cada estado com um novo campo que registe os estados usados para gerar cada constituinte. Assim, o procedimento COMPLETER adiciona um ponteiro, para o estado antecessor, à lista dos estados antecessores.

Chart[0]

S0	δ →•F	[0,0]	[]
S1	F →•SN SV	[0,0]	[]
S2	SN→•SNA	[0,0]	[]
S3	SN→•SNA SP	[0,0]	[]
S4	SNA→•Det N	[0,0]	[]
S5	SNA→•Det N Adj	[0,0]	[]
S6	SNA→•Det NP	[0,0]	[]

Chart[1]

S7	Det→os•	[0,1]	[]
S8	SNA→Det•N	[0,1]	[S7]
S9	SNA→Det•N Adj	[0,1]	[S7]
S10	SNA→Det•NP	[0,1]	[S7]

Chart[2]

S11	N→professores•	[1,2]	[]
S12	SNA→Det N•	[0,2]	[S7,S11]
S13	SNA→Det N•Adj	[0,2]	[S7,S11]
S14	SN→SNA•	[0,2]	[S12]
S15	SN→SNA•SP	[0,2]	[S12]
S16	F →SN•SV	[0,2]	[S14]
S17	SP→•P SN	[2,2]	[]
S18	SV→•V	[2,2]	[]
S19	SV→•V SN	[2,2]	[]

Chart[3]

S20	V→dão•	[2,3]	[]
S21	SV→V•	[2,3]	[S20]
S22	SV→V•SN	[2,3]	[S20]
S23	SN→•SNA	[3,3]	[]
S24	SN→•SNA SP	[3,3]	[]
S25	SNA→•Det N	[3,3]	[]
S26	SNA→•Det N Adj	[3,3]	[]
S27	SNA→•Det NP	[3,3]	[]

Chart[4]

S28	Det→as•	[3,4]	[]
S29	SNA→Det•N	[3,4]	[S28]
S30	SNA→Det•N Adj	[3,4]	[S28]
S31	SNA→Det•NP	[3,4]	[S28]

Chart[5]

S32	N→aulas•	[4,5]	[]
S33	SNA→Det N•	[3,5]	[S28,S32]
S34	SNA→Det N•Adj	[3,5]	[S28,S32]
S35	SN→SNA•	[3,5]	[S33]
S36	SN→SNA•SP	[3,5]	[S33]
S37	SV→V SN•	[2,5]	[S35]
S38	F →SN SV•	[0,5]	[S14,S37]
S39	δ →F•	[0,5]	[S16]
S40	SP→•P SN	[5,5]	[]

Chart[6]

S41	P→em•	[5,6]	[]
S42	SP→P•SN	[5,6]	[S41]
S43	SN→•SNA	[6,6]	[]
S44	SN→•SNA SP	[6,6]	[]
S45	SNA→•Det N	[6,6]	[]
S46	SNA→•Det N Adj	[6,6]	[]
S47	SNA→•Det NP	[6,6]	[]

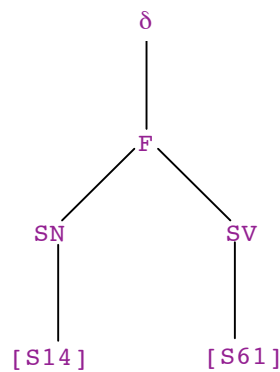
Chart[7]

S48	Det→as•	[6,7]	[]
S49	SNA→Det•N	[6,7]	[S48]
S50	SNA→Det•N Adj	[6,7]	[S48]
S51	SNA→Det•NP	[6,7]	[S48]

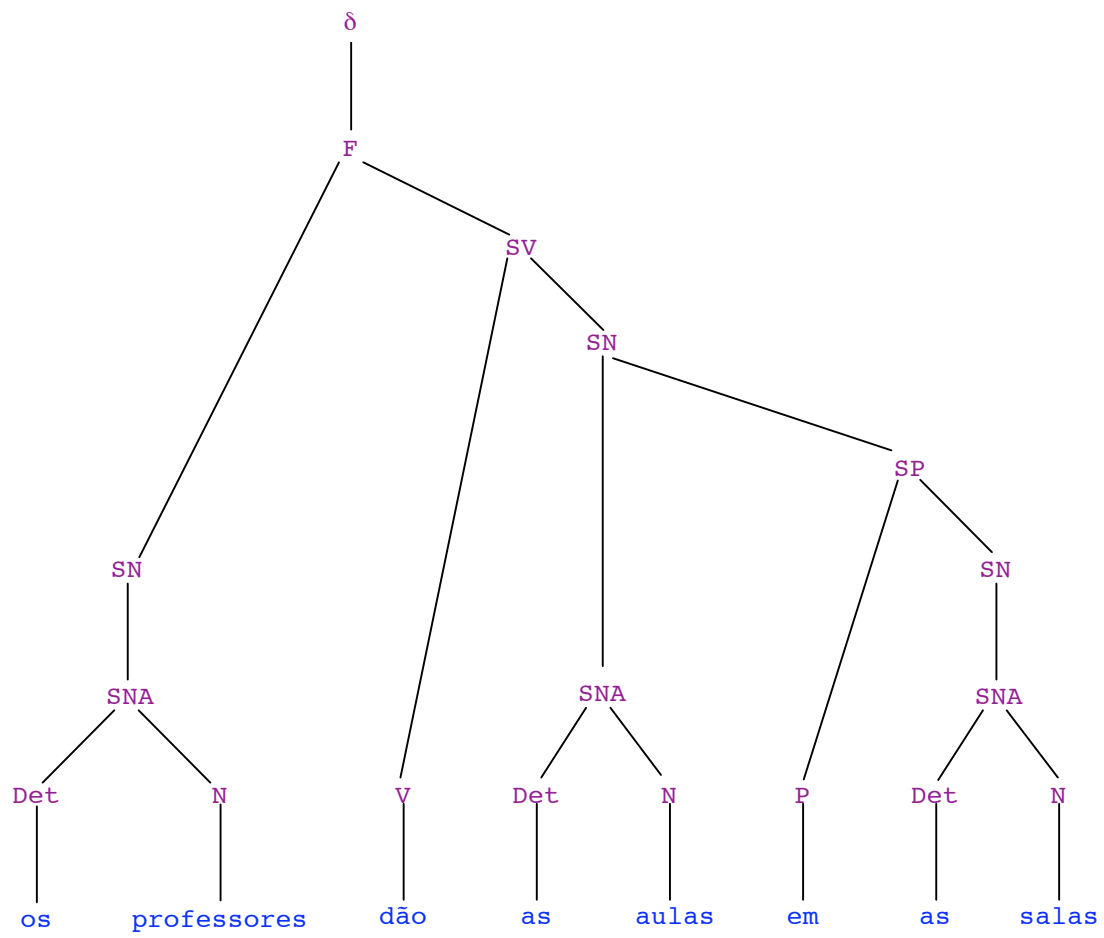
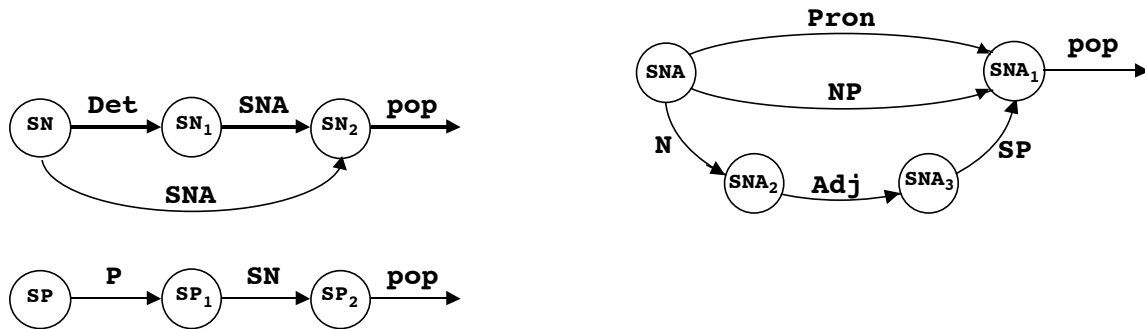
Chart[8]

S52	N→salas•	[7,8]	[]
S53	SNA→Det N•	[6,8]	[S48,S52]
S54	SNA→Det N•Adj	[6,8]	[S48,S52]
S55	SN→SNA•	[6,8]	[S53]
S56	SN→SNA•SP	[6,8]	[S53]
S57	SN→SNA•	[6,8]	[S55]
S58	SN→SNA•SP	[6,8]	[S55]
S59	SP→P SN•	[5,8]	[S41,S57]
S60	SN→SNA SP•	[3,8]	[S33,S59]
S61	SV→V SN•	[2,8]	[S20,S60]
S62	F →SN SV•	[0,8]	[S14,S61]
S63	δ →F•	[0,8]	[S62]
S64	SP→•P SN	[8,8]	[]

No estado S63 derivou-se o símbolo original usando todas as palavras da frase. O estado S39 também deriva o símbolo original mas só faz uso das 5 primeiras palavras ("Os professores dão as aulas").

A estrutura adicionada a cada estado pode agora ser usada para determinar a estrutura da análise efectuada com sucesso. Assim, verifica-se que o símbolo inicial se obteve usando o estado S62, que por sua vez se obteve usando a regra F→SN SV e os estados S14 e S61, ou seja, SN foi derivado no estado S14 e SV foi derivado no estado S61. Graficamente, pode-se representar assim, a sub-árvore já identificada:

O constituinte SN foi identificado através da regra SN→SNA e do constituinte obtido no estado S12, que por sua vez foi obtido através da regra SNA→Det N e dos constituintes S7 e S11, respectivamente. S7 corresponde à palavra os (um Det) e S11 corresponde à palavra professores (um N). De modo análogo pode-se identificar a constituição do SV (estado S61).

A árvore correspondente é a seguinte:

δ
F
SV
SN
SP
SN
SN
SNA
SNA
SNA
Det
N
V
Det
N
P
Det
N
os
professores
dão
as
aulas
em
as
salas

Problema 24 — Análise (rede de transições) de "*Um carro amarelo em o domingo*"

1 um 2 carro 3 amarelo 4 em 5 o 6 Domingo 7

	NA	PA	ARC	Pilha Pontos de Retorno	Estados Alternativos
1	SN	1	Det	()	**(SN,1,SNA,())**
2	SN_1	2	SNA	()	*…sem alteração.*
3	SNA	2	N	(SN_2)	*…sem alteração.*
4	SNA_2	3	Adj	(SN_2)	*…sem alteração.*
5	SNA_3	4	SP	(SN_2)	*…sem alteração.*
6	SP	4	P	(SNA_1,SN_2)	*…sem alteração.*
7	SP_1	5	SN	(SNA_1,SN_2)	*…sem alteração.*
8	SN	5	Det	(SNA_1,SN_2)	**(SN,5,SNA,(SP_2,SN_2,SNA_1))** (SN,1,SNA,())
9	SN_1	6	SNA	(SP_2,SNA_1,SN_2)	*…sem alteração.*
10	SNA	6	NP	(SN_2,SP_2,SNA_1,SN_2)	**(SNA,6,N,(SN_2,SP_2,SN_2,SNA_1))** (SN,5,SNA,(SP_2,SN_2,SNA_1)) (SN,1,SNA,())
11	SNA_1	7	pop	(SN_2,SP_2,SNA_1,SN_2)	*…sem alteração.*
12	SN_2	7	pop	(SP_2,SNA_1,SN_2)	*…sem alteração.*
13	SP_2	7	pop	(SNA_1,SN_2)	*…sem alteração.*
14	SNA_1	7	pop	(SN_2)	*…sem alteração.*
15	SN_2	7	pop	()	*…sem alteração.*

Sucesso!

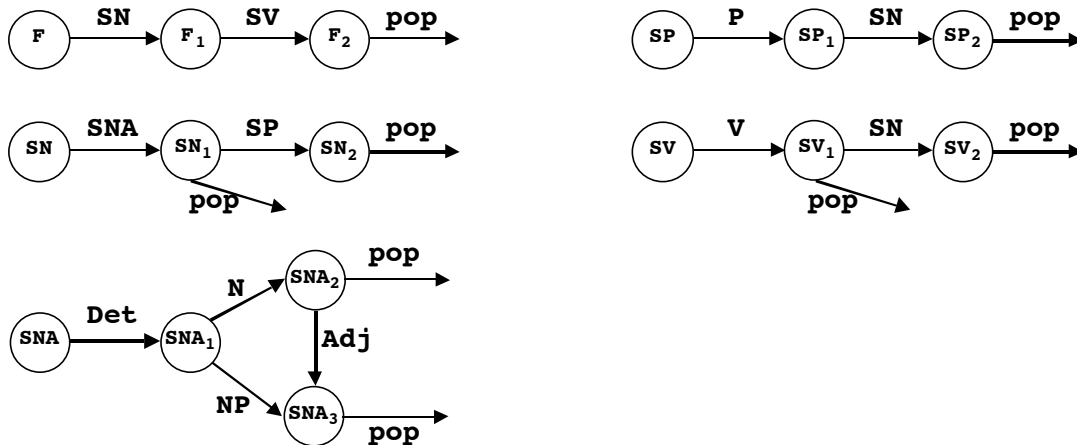
Nota: incluem-se na lista de estados alternativos todos os estados correspondentes a arcos ainda não explorados (independentemente de poderem ou não ser explorados). Também se considera que os arcos são explorados pela ordem porque aparecem representados na rede, ou seja de cima para baixo.

Problema 25 — Análise (rede de transições) de "*O Zé Maria ganhou o concurso BigBrother*"

1 O 2 Zé 3 Maria 4 ganhou 5 o 6 concurso 7 BigBrother 8

	NA	PA	ARC	Pilha Pontos de Retorno	Estados Alternativos
1	F	1	SN	()	
2	SN	1	Det	(F_1)	
3	SN_1	2	NP	(F_1)	
4	SN_2	3	JUMP	(F_1)	**(SN_2,3,NP,(F_1))**.
5	SN_3	3	pop	(F_1)	*...sem alteração.*
6	F_1	3	SV	()	*...sem alteração.*
7	SV	3	V	(F_2)	*...sem alteração.*
			fail!		
8	SN_2	3	NP	(F_1)	
9	SN_3	4	pop	(F_1)	
10	F_1	4	SV	()	
11	SV	4	V	(F_2)	
12	SV_1	5	SN	(F_2)	
13	SN	5	Det	(SV_2,F_2)	
14	SN_1	6	N	(SV_2,F_2)	**(SN_1,6,NP,(SV_2,F_2))**
15	SN_2	7	JUMP	(SV_2,F_2)	**(SN_2,7,NP,(SV_2,F_2))** (SN_1,6,NP,(SV_2,F_2))
16	SN_3	7	pop	(SV_2,F_2)	*...sem alteração.*
17	SV_2	7	pop	(F_2)	*...sem alteração.*
18	F_2	7	pop	()	*...sem alteração.*
			fail!		
19	SN_2	7	NP	(SV_2,F_2)	**(SN_1,6,NP,(SV_2,F_2))**
20	SN_3	8	pop	(SV_2,F_2)	*...sem alteração.*
21	SV_2	8	pop	(F_2)	*...sem alteração.*
22	F_2	8	pop	()	*...sem alteração.*

Sucesso!

Problema 26 — Análise (rede de transições) de "*Os professores dão as aulas em as salas*"

$_1$ Os $_2$ professores $_3$ dão $_4$ as $_5$ aulas $_6$ em $_7$ as $_8$ salas $_9$

	NA	PA	ARC	Pilha Pontos de Retorno	Estados Alternativos
1	F	1	SN	()	
2	SN	1	SNA	(F_1)	
5	SNA	1	Det	(SN_1,F_1)	
3	SNA_1	2	N	(SN_1,F_1)	**(SNA_1,2,NP,(SN_1,F_1))**
4	SNA_2	3	pop	(SN_1,F_1)	**(SNA_2,3,Adj,(SN_1,F_1))** (SNA_1,2,NP,(SN_1,F_1))
5	SN_1	3	SP	(F_1)	**(SN_1,3,pop,(F_1))** (SNA_2,3,Adj,(SN_1,F_1)) (SNA_1,2,NP,(SN_1,F_1))
6	SP	3	**fail!**	(SN_2,F_1)	*...sem alteração.*
7	SN_1	3	pop	(F_1)	(SNA_2,3,Adj,(SN_1,F_1)) (SNA_1,2,NP,(SN_1,F_1))
8	F_1	3	SV	()	*...sem alteração.*
9	SV	3	V	(F_2)	*...sem alteração.*
10	SV_1	4	SN	(F_2)	**(SV_1,4,pop,(F_2))** (SNA_2,3,Adj,(SN_1,F_1)) (SNA_1,2,NP,(SN_1,F_1))
11	SN	4	SNA	(SV_2,F_2)	*...sem alteração.*
12	SNA	4	Det	(SN_1,SV_2,F_2)	*...sem alteração.*
13	SNA_1	5	N	(SN_1,SV_2,F_2)	**(SNA_1,5,NP,(SN_1,SV_2,F_2))** (SV_1,4,pop,(F_2)) (SNA_2,3,Adj,(SN_1,F_1)) (SNA_1,2,NP,(SN_1,F_1))
14	SNA_2	6	pop	(SN_1,SV_2,F_2)	**(SNA_2,6,Adj,(SN_1,SV_2,F_2))** (SNA_1,5,NP,(SN_1,SV_2,F_2)) (SV_1,4,pop,(F_2)) (SNA_2,3,Adj,(SN_1,F_1)) (SNA_1,2,NP,(SN_1,F_1))
15	SN_1	6	SP	(SV_2,F_2)	**(SN_1,6,pop,(SV_2,F_2))** (SNA_2,6,Adj,(SN_1,SV_2,F_2)) (SNA_1,5,NP,(SN_1,SV_2,F_2)) (SV_1,4,pop,(F_2)) (SNA_2,3,Adj,(SN_1,F_1)) (SNA_1,2,NP,(SN_1,F_1))
16	SP	6	P	(SN_2,SV_2,F_2)	*...sem alteração.*

	NA	PA	ARC	Pilha Pontos de	Estados Alternativos

				Retorno	
16	SP	6	P	(SN_2,SV_2,F_2)	...*sem alteração.*
17	SP_1	7	SN	(SN_2,SV_2,F_2)	...*sem alteração.*
18	SN	7	SNA	(SP_2,SN_2,SV_2,F_2)	...*sem alteração.*
19	SNA	7	Det	$(SN_1,SP_2,SN_2,SV_2,F_2)$	...*sem alteração.*
20	SNA_1	8	N	$(SN_1,SP_2,SN_2,SV_2,F_2)$	**$(SNA_1,8,NP,(SN_2,SP_2,SN_2,SV_2,F_2))$** $(SN_1,6,pop,(SV_2,F_2))$ $(SNA_2,6,Adj,(SN_1,SV_2,F_2))$ $(SNA_1,5,NP,(SN_1,SV_2,F_2))$ $(SV_1,4,pop,(F_2))$ $(SNA_2,3,Adj,(SN_1,F_1))$ $(SNA_1,2,NP,(SN_1,F_1))$
21	SNA_2	9	pop	$(SN_1,SP_2,SN_2,SV_2,F_2)$	...*sem alteração.*
22	SN_1	9	SP	(SP_2,SN_2,SV_2,F_2)	**$(SN_1,9,pop,(SP_2,SN_2,SV_2,F_2))$** $(SNA_1,8,NP,(SN_2,SP_2,SN_2,SV_2,F_2))$ $(SN_1,6,pop,(SV_2,F_2))$ $(SNA_2,6,Adj,(SN_1,SV_2,F_2))$ $(SNA_1,5,NP,(SN_1,SV_2,F_2))$ $(SV_1,4,pop,(F_2))$ $(SNA_2,3,Adj,(SN_1,F_1))$ $(SNA_1,2,NP,(SN_1,F_1))$
23	SP	9	**fail!**	$(SN_2,SP_2,SN_2,SV_2,F_2)$	...*sem alteração.*
24	SN_1	9	pop	(SP_2,SN_2,SV_2,F_2)	**$(SNA_1,8,NP,(SN_2,SP_2,SN_2,SV_2,F_2))$** $(SN_1,6,pop,(SV_2,F_2))$ $(SNA_2,6,Adj,(SN_1,SV_2,F_2))$ $(SNA_1,5,NP,(SN_1,SV_2,F_2))$ $(SV_1,4,pop,(F_2))$ $(SNA_2,3,Adj,(SN_1,F_1))$ $(SNA_1,2,NP,(SN_1,F_1))$
25	SP_2	9	pop	(SN_2,SV_2,F_2)	...*sem alteração.*
26	SN_2	9	pop	(SV_2,F_2)	...*sem alteração.*
27	SV_2	9	pop	(F_2)	...*sem alteração.*
28	F_2	9	pop	$()$	...*sem alteração.*

Sucesso!

Problema 27 — Frases aceites por uma rede de transições.

Léxico:

o:	Det	azul:	Adj
a:	Det	dourada:	Adj
os:	Det	aleijada:	Adj
pardal:	N	comeu:	VT
pevide:	N	comprar:	VT
pata:	N	deu:	VT
João:	NP	dormia:	VI
Maria:	NP	brilhava:	VI
Lisboa:	NP	pensava:	VI

Frases aceites:

- O pardal azul comeu a pevide que brilhava.
- O pardal azul comeu a pevide dourada.
- O pardal que dormia deu a pata aleijada.
- O pardal que comeu a pevide dourada dormia.
- O pardal azul pensava.

Problema 28 — Análise (rede de transições estendida) de "*O prof viu o gato*"

Para se perceber melhor a interacção entre as diferentes redes, apresenta-se a solução separando o processamento associado a cada uma das redes. Posteriormente, apresenta-se a mesma solução, mas usando a ordem cronológica do algoritmo.

Considera-se que os arcos pop, quando atravessados, devolvem uma estrutura com todos os registos que estejam associados com a rede que esteve a ser analisada.

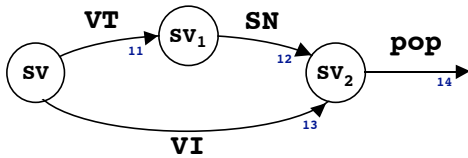
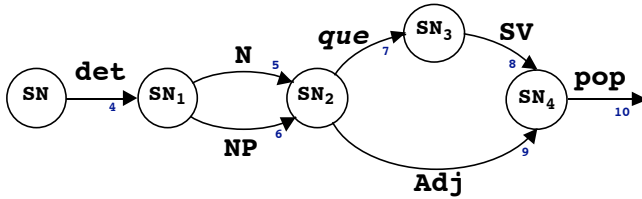
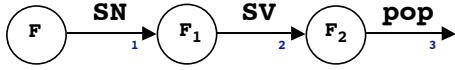
$_1$ O $_2$ prof $_3$ viu $_4$ o $_5$ gato $_6$

	NA	PA	ARC	Registos	Pilha PR	E Alternativos
1	F	1	SN		( )	
5	F_1	3	V {m}∩{m f} {s}∩{s}	**SUJ** ← (SN **DET** o **N** prof **GEN** m **NUM** s) **VERBO** ← ver **GEN** ← m **NUM** ← s		*…sem alteração.*
6	F_2	4	SN		( )	*…sem alteração.*
10	F_3	6	pop	**COMPL** ←(SN **DET** o N gato **GEN** m **NUM** s)	( )	*…sem alteração.*
				(F **SUJ** (SN **DET** o **N** prof **GEN** m **NUM** s) **VERBO** ver **GEN** m **NUM** s **COMPL** (SN **DET** o **N** gato **GEN** m **NUM** s)))		
2	SN	1	Det		(F_1)	*…sem alteração.*
3	SN_1	2	N {m}∩{m,f} {s}∩{s}	**DET** ← o **GEN** ← m **NUM** ← s	(F_1)	*…sem alteração.*
4	SN_2	3	pop	**N** ← prof	(F_1)	*…sem alteração.*
				(SN **DET** o **N** prof **GEN** m **NUM** s)		
7	SN	4	Det		(F_3)	*…sem alteração.*
8	SN_1	5	N {m}∩{m} {s}∩{s}	**DET** ← o **GEN** ← m **NUM** ← s	(F_3)	*…sem alteração.*
9	SN_2	6	pop	**N** ← gato	(F_3)	*…sem alteração.*
				(SN **DET** o **N** gato **GEN** m **NUM** s)		

Sucesso!

	NA	PA	ARC	Registos	Pilha PR	E Alternativos
1	F	1	SN		( )	
2	SN	1	Det		(F_1)	…sem alteração.
3	SN_1	2	N {m}∩{m,f} {s}∩{s}	**DET** ← o **GEN** ← m **NUM** ← s	(F_1)	…sem alteração.
4	SN_2	3	pop	**N** ← prof	(F_1)	…sem alteração.
				(SN **DET** o **N** prof **GEN** m **NUM** s)		
5	F_1	3	V {m}∩{m f} {s}∩{s}	**SUJ** ← (SN **DET** o **N** prof **GEN** m **NUM** s) **VERBO** ← ver **GEN** ← m **NUM** ← s		…sem alteração.
6	F_2	4	SN		( )	…sem alteração.
7	SN	4	Det		(F_3)	…sem alteração.
8	SN_1	5	N {m}∩{m} {s}∩{s}	**DET** ← o **GEN** ← m **NUM** ← s	(F_3)	…sem alteração.
9	SN_2	6	pop	**N** ← gato	(F_3)	…sem alteração.
				(SN **DET** o **N** gato **GEN** m **NUM** s)		
10	F_3	6	pop	**COMPL** ←(SN **DET** o N gato **GEN** m **NUM** s)	( )	…sem alteração.
				(F **SUJ** (SN **DET** o **N** prof **GEN** m **NUM** s) **VERBO** ver **GEN** m **NUM** s **COMPL** (SN **DET** o **N** gato **GEN** m **NUM** s)))		

Sucesso!

Problema 29 — Análise (rede de transições estendida) de "*O pardal azul comeu a pevide que brilhava*"

Arco	Teste	Acções
1	–	–
2	–	–
3	–	–
4	–	**GEN** ← **GEN**$_*$
5	**GEN** ∩ **GEN**$_*$	**COM** ← **COM**$_*$ **GEN** ← **GEN**∩**GEN**$_*$
6	**GEN** ∩ **GEN**$_*$	**COM** ← **COM**$_*$ **GEN** ← **GEN**∩**GEN**$_*$
7	–	–
8	–	–
9	**GEN** ∩ **GEN**$_*$	**COM** ← **COM**$_*$ **GEN** ← **GEN**∩**GEN**$_*$
10	–	–
11	–	**COM** ← **COM**$_*$ **VT** ← {}
12	–	–
13	–	**VI** ← {s}
14	**VI** ∨ **COM**∩**COM**$_{SN}$	–

Léxico actualizado:

o: Det **RAIZ** o **GEN** m
a: Det **RAIZ** o **GEN** f
os: Det **RAIZ** o **GEN** m
pardal: N **RAIZ** pardal **GEN** m **COM** n
pevide: N **RAIZ** pevide **GEN** f **COM** s
pata: N **RAIZ** pata **GEN** f **COM** s
João: NP **RAIZ** João **GEN** m **COM** n
Maria: NP **RAIZ** Maria **GEN** f **COM** n
Lisboa: NP **RAIZ** Lisboa **GEN** f **COM** n
azul: Adj **RAIZ** azul **GEN** {m f}
dourada: Adj **RAIZ** dourado **GEN** f
aleijada: Adj **RAIZ** Joaleijar **GEN** f
comeu: VT **RAIZ** comer **COM** s
bebia: VT **RAIZ** beber **COM** s
comprar: VT **RAIZ** comprar **COM** {s n}
deu: VT **RAIZ** dar **COM** {s n}
dormia: VI **RAIZ** dormir
brilhava: VI **RAIZ** brilhar
pensava: VI **RAIZ** pensar

Para se perceber melhor a interacção entre as diferentes redes, apresenta-se a solução separando o processamento associado a cada uma das redes. Posteriormente, apresenta-se a mesma solução, mas usando a ordem cronológica do algoritmo.

Os arcos pop, quando atravessados, devolvem uma estrutura com todos os registos que estejam associados com a rede que esteve a ser analisada.

$_1$ O $_2$ pardal $_3$ azul $_4$ comeu $_5$ a $_6$ pevide $_7$ que $_8$ brilhava $_9$

	NA	PA	ARC	Testes	Registos	Pilha PR	E Alternativos
1	F	1	SN			()	
6	F_1	4	SV		(SN (**GEN** {m}) (**COM** {n}))	()	*…sem alteração.*
17	F_2	9	pop		(SN (**GEN** {m}) (**COM** {n})) (SV (**COM** {s}) (**VI** {}) (SN (**GEN** {f}) (**COM** {s}) (SV (**VI**{s}))))	()	*…sem alteração.*
2	SN	1	Det			(F_1)	
3	SN_1	2	N	{m}∩{m}	**GEN** ← {m}	(F_1)	**(SN_1,2,NP,(F_1),(GEN:=m))**
4	SN_2	3	Adj	{m} ∩ {m f}	**GEN** ← {m} **COM** ← {n}	(F_1)	*…sem alteração.*
5	SN_4	4	pop		*…sem alteração.*	(F_1)	*…sem alteração.*
7	SV	4	VT			(F_2)	**(SV,4,VI,(F_2),())** (SN_1,2,NP,(F_1),(GEN=m))
8	SV_1	5	SN		**VI** ← {} **COM** ← {s}	(F_2)	*…sem alteração.*
16	SV_2	9	pop	{} v {s}∩{s}	**COM** ← {s} **VI** ← {} (SN (**GEN** {f}) (**COM** {s}) (SV (**VI**{s})))	(F_2)	*…sem alteração.*
9	SN	5	Det			(SV_2,F_2)	*…sem alteração.*
10	SN_1	6	N	{f}∩{f}	**GEN** ← {f}	(SV_2,F_2)	**(SN_1,6,NP,(SV_2,F_2),(GEN=f))** (SV,4,VI,(F_2),()) (SN_1,2,NP,(F_1),(GEN=m))
11	SN_2	7	que		**GEN** ← {f} **COM** ← {s}	(SV_2,F_2)	**(SN_2,7,Adj,(SV_2,F_2),(GEN=f, COM=s))** (SN_1,6,NP,(SV_2,F_2),(GEN=f)) (SV,4,VI,(F_2),()) (SN_1,2,NP,(F_1),(GEN=m))
12	SN_3	8	SV		…sem alteração.	(SV_2,F_2)	*…sem alteração.*
15	SN_4	9	pop		**GEN** ← {f} **COM** ← {s} (SV (**VI** s))	(SV_2,F_2)	*…sem alteração.*
13	SV	8	VI			(SN_4,SV_2,F_2)	*…sem alteração.*
14	SV_2	9	pop	s v ?∩?	**VI** ← {s}	(SN_4,SV_2,F_2)	*…sem alteração.*

Sucesso!

1 O 2 pardal 3 azul 4 comeu 5 a 6 pevide 7 que 8 brilhava 9

	NA	PA	ARC	Testes	Registos	Pilha PR	E Alternativos
1	F	1	SN			()	
2	SN	1	Det			(F$_1$)	
3	SN$_1$	2	N	{m}∩{m}	**GEN** ← {m}	(F$_1$)	**(SN$_1$,2,NP,(F$_1$),(GEN:=m))**
4	SN$_2$	3	Adj	{m} ∩ {m f}	**GEN** ← {m} **COM** ← {n}	(F$_1$)	*...sem alteração.*
5	SN$_4$	4	pop		*...sem alteração.*	(F$_1$)	*...sem alteração.*
6	F$_1$	4	SV		(SN (**GEN** {m}) (**COM** {n}))	()	*...sem alteração.*
7	SV	4	VT			(F$_2$)	**(SV,4,VI,(F$_2$),())** (SN$_1$,2,NP,(F$_1$),(GEN=m))
8	SV$_1$	5	SN		**VI** ← {} **COM** ← {s}	(F$_2$)	*...sem alteração.*
9	SN	5	Det			(SV$_2$,F$_2$)	*...sem alteração.*
10	SN$_1$	6	N	{f}∩{f}	**GEN** ← {f}	(SV$_2$,F$_2$)	**(SN$_1$,6,NP,(SV$_2$,F$_2$),(GEN=f))** (SV,4,VI,(F$_2$),()) (SN$_1$,2,NP,(F$_1$),(GEN=m))
11	SN$_2$	7	que		**GEN** ← {f} **COM** ← {s}	(SV$_2$,F$_2$)	**(SN$_2$,7,Adj,(SV$_2$,F$_2$),(GEN=f, COM=s))** (SN$_1$,6,NP,(SV$_2$,F$_2$),(GEN=f)) (SV,4,VI,(F$_2$),()) (SN$_1$,2,NP,(F$_1$),(GEN=m))
12	SN$_3$	8	SV		*...sem alteração.*	(SV$_2$,F$_2$)	*...sem alteração.*
13	SV	8	VI			(SN$_4$,SV$_2$,F$_2$)	*...sem alteração.*
14	SV$_2$	9	pop	s v ?∩?	**VI**:= {s}	(SN$_4$,SV$_2$,F$_2$)	*...sem alteração.*
15	SN$_4$	9	pop		**GEN** ← {f} **COM** ← {s} (SV (**VI** s))	(SV$_2$,F$_2$)	*...sem alteração.*
16	SV$_2$	9	pop	{} v {s}∩{s}	**COM** ← {s} **VI** ← {} (SN (**GEN** {f}) (**COM** {s}) (SV (**VI**{s}))))	(F$_2$)	*...sem alteração.*
17	F$_2$	9	pop		(SN (**GEN** {m}) (**COM** {n})) (SV (**COM** {s}) (**VI** {}) (SN (**GEN** {f}) (**COM** {s}) (SV (**VI**{s})))	()	*...sem alteração.*

Sucesso!

Problema 30 — Reconhecedor de números usando o Prolog (DCG)

```
número(0) --> "zero".
número(N) --> converte(N).

converte(100) --> "cem".
converte(N) --> centenas(N).
converte(N) --> centenas(C),espaço,"e",espaço,converte_resto(R),{N is C+R}.
converte(N) --> converte_resto(N).

converte_resto(N) --> unidades(N).
converte_resto(N) --> dezenas(N).
converte_resto(N) --> irregulares(N).
converte_resto(N) --> dezenas(D),espaço,"e",espaço,unidades(U),{N is D+U}.

espaço --> " ".
espaço --> " ", espaço.

centenas(100) --> "cento".
centenas(200) --> "duzentos".
centenas(300) --> "trezentos".
centenas(400) --> "quatrocentos".
centenas(500) --> "quinhentos".
centenas(600) --> "seiscentos".
centenas(700) --> "setecentos".
centenas(800) --> "oitocentos".
centenas(900) --> "novecentos".

dezenas(20) --> "vinte".
dezenas(30) --> "trinta".
dezenas(40) --> "quarenta".
dezenas(50) --> "cinquenta".
dezenas(60) --> "sessenta".
dezenas(70) --> "setenta".
dezenas(80) --> "oitenta".
dezenas(90) --> "noventa".

irregulares(10) --> "dez".
irregulares(11) --> "onze".
irregulares(12) --> "doze".
irregulares(13) --> "treze".
irregulares(14) --> "catorze".
irregulares(15) --> "quinze".
irregulares(16) --> "dezasseis".
irregulares(17) --> "dezassete".
irregulares(18) --> "dezoito".
irregulares(19) --> "dezanove".

unidades(1) --> "um".
unidades(2) --> "dois".
unidades(3) --> "três".
unidades(4) --> "quatro".
unidades(5) --> "cinco".
unidades(6) --> "seis".
unidades(7) --> "sete".
unidades(8) --> "oito".
unidades(9) --> "nove".

?- número(N, "seiscentos e quarenta e     nove", [])
Nº1    N = 649
No more solutions

?- número(N, "seiscentos e quarenta", [])
Nº1    N = 640
No more solutions

?- número(N, "seiscentos e três", [])
Nº1    N = 603
No more solutions
```

Problema 31 — Análise de "*Os professores dão as aulas em as salas*" em Prolog (DCG)

Léxico:

```
det(det(o))     --> "as".
det(det(o))     --> "os".
pron(pron(o))   --> "as".
pron(pron(o))   --> "os".
n(n(aula))      --> "aulas".
n(n(prof))      --> "professores".
n(n(sala))      --> "salas".
v(v(dar))       --> "dão".
p(p(em))        --> "em".
np(np(nuno))    --> "Nuno".
adj(adj(azul)) --> "azul".
```

Gramática:

```
f(f(Asn,Asv))    --> sn(Asn),espaço,sv(Asv).

sn(sn(Asna))      --> sna(Asna).
sn(sn(Asna,Asp))  --> sna(Asna),espaço,sp(Asp).
sna(sna(Adet,An))  --> det(Adet),espaço,n(An).
sna(sna(Adet,Anp)) --> det(Adet),espaço,np(Anp).
sna(sna(Adet,An,Aadj)) --> det(Adet),espaço,n(An),espaço,adj(Aadj).

sp(sp(Ap,Asn)) --> p(Ap),espaço,sn(Asn).

sv(sv(Av))     --> v(Av).
sv(sv(Av,Asn)) --> v(Av),espaço,sn(Asn).

espaço --> " ".
espaço --> " ",espaço.
```

Análise sintáctica:

```
?- f(Arvore,"os professores dão as aulas em as salas",[]).

Nº1  Arvore =  f(sn(sna(det(o),
                       n(prof))),
                sv(v(dar),
                   sn(sna(det(o),n(aula)),sp(p(em),
                                              sn(sna(det(o),n(sala))))))
No more solutions
```

Problema 32 — Verificação da sintaxe usando o Prolog (DCG)

```
frase --> sn,espaço,sv.
sv --> verbo,espaço,sn.
sn --> nome.
nome --> nome_comum.
nome --> nome_próprio.

espaço --> " ".
espaço --> " ",espaço.

verbo --> "tem".
verbo --> "há".
verbo --> "come".

nome_comum --> "hotéis".
nome_comum --> "piscinas".
nome_comum --> "javalis".

nome_próprio --> "Lisboa".
nome_próprio --> "Porto".
nome_próprio --> "Portugal".


?-  frase("Portugal tem javalis",[])
yes

?-  frase("Portugal javalis tem",[])
no
```

Problema 33 — Verificação da sintaxe de 5 frases usando o Prolog (DCG)

Léxico:

```
det(det(o)) --> "o".
det(det(o)) --> "O".
det(det(o)) --> "a".
det(det(o)) --> "as".

n(n(pardal)) --> "pardal".
n(n(pevide)) --> "pevide".
n(n(pata))   --> "pata".

vt(vt(dar))  --> "deu".
```

```
vt(vt(comer)) --> "comeu".
vi(vi(dormir)) --> "dormia".
vi(vi(brilhar))--> "brilhava".
vi(vi(pensar)) --> "pensava".

np(np(nuno)) --> "Nuno".

adj(adj(azul))    --> "azul".
adj(adj(dourada)) --> "dourada".
adj(adj(aleijada))--> "aleijada".
```

Gramática:

```
f(f(Asn,Asv)) --> sn(Asn),espaço,sv(Asv).

sv(sv(Avt,Asn)) --> vt(Avt),espaço,sn(Asn).
sv(sv(Avi))     --> vi(Avi).

sn(sn(Adet,Asna,Aadj)) --> det(Adet),espaço,sna(Asna),espaço,adj(Aadj).
sn(sn(Adet,Asna,que,Asv)) -->  det(Adet),espaço,sna(Asna),
                               espaço,"que",espaço,sv(Asv).

sna(An)  --> n(An).
sna(Anp) --> np(Anp).

espaço --> " ".
espaço --> " ",espaço.
```

Análise sintáctica:

```
?- f(Arvore,"O pardal azul comeu a pevide que brilhava",[]).
Nº1  Arvore =  f(sn(det(o),n(pardal),adj(azul)),
                 sv(vt(comer),
                    sn(det(o),n(pevide),que,sv(vi(brilhar)))))
No more solutions

?- f(Arvore,"O pardal azul comeu a pevide dourada",[]).
Nº1  Arvore =  f(sn(det(o),n(pardal),adj(azul)),
                 sv(vt(comer),
                    sn(det(o),n(pevide),adj(dourada))))
No more solutions

?- f(Arvore,"O pardal que dormia deu a pata aleijada",[]).
Nº1  Arvore =  f(sn(det(o),n(pardal),que,sv(vi(dormir))),
                sv(vt(dar),
                   sn(det(o),n(pata),adj(aleijada))))
No more solutions

?- f(Arvore,"O pardal que comeu a pevide dourada dormia",[]).
Nº1  Arvore =  f(sn(det(o),n(pardal),que,sv(vt(comer),
                                            sn(det(o),n(pevide),adj(dourada)))),
                 sv(vi(dormir)))
No more solutions

?- f(Arvore,"O pardal azul pensava",[]).
Nº1  Arvore =  f(sn(det(o),n(pardal),adj(azul)),
                 sv(vi(pensar)))
No more solutions
```

Problema 34 — Conversão de uma frase para uma linguagem lógica com Prolog (DCG)

```
converte -->  nome_próprio(Np),espaço,verbo([V1,V2,V3]),
              espaço,nome_comum([Nc1,Nc2]),
              {G1=..[V1,Np,Nc2],G2=..[Nc1,Nc2],display(G1),nl,display(G2),nl}.

verbo([ter,X,Y])   --> "tem".
verbo([haver,X,Y]) --> "há".
verbo([comer,X,Y]) --> "come".

nome_comum([hotel,X])  --> "hotéis".
nome_comum([piscina,X])--> "piscinas".
nome_comum([javali,X]) --> "javalis".

nome_próprio(lisboa)  --> "Lisboa".
nome_próprio(porto)   --> "Porto".
nome_próprio(portugal)--> "Portugal".

?- converte("Porto come javalis",[])
comer(porto,_1037)
javali(_1037)
Nº1    yes
No more solutions

?- converte("Lisboa tem hotéis",[])
ter(lisboa,_1093)
hotel(_1093)
Nº1    yes
No more solutions
```

Problema 35 — Representação na forma lógica e na forma quase-lógica

- `Todos os homens amam uma mulher`

Forma lógica:

```
(TODO h1: (HOMEM1 h1)
          (UM m1: (MULHER1 m1)
                  (AMA1 h1 m1)))

(UM m2: (MULHER1 m2)
        (TODO h2: (HOMEM1 h2)
                  (AMA1 h2 m2)))
```

Forma quase-lógica: `(AMA1 <TODO h3: HOMEM1> <UM m3: MULHER1>)`

- `A maioria dos homens compram uma televisão`

Forma lógica:

```
(MAIORIA h4: (HOMEM1 h4)
             (UM t1: (TV1 t1)
                     (COMPRA1 h4 t1)))

(UM t2: (TV1 t2)
        (MAIORIA h5: (HOMEM1 h5)
                     (COMPRA1 h5 t2)))
```

Forma quase-lógica: `(COMPRA1 <MAIORIA h6: HOMEM1> <UM t3: TV1>)`

- `A maioria dos cães ladra`

Forma lógica: `(MAIORIA c1: (CÃO1 c1) (LADRA1 c1))`

~~`(MAIORIA c2: (LADRA1 c2) (CÃO1 c2))`~~ Não é aceitável pois teria a interpretação de que “a maioria das coisas que ladram são cães”.

Forma quase-lógica: `(LADRA1 <MAIORIA c3: CÃO1>)`

- `A Susana viu a chave`

Forma lógica: `(O c4: (CHAVE1 c4) (VER1 “Susana” c4))`
(Utensílio para abrir a porta)

`(O c5: (CHAVE2 c5) (VER1 “Susana” c5))`
(Conjunto de números do totoloto)

Forma quase-lógica: `(VER1 (NOME s1 “Susana”) <UM c6: {CHAVE1 CHAVE2 }>)`

Problema 36 — Representação na forma quase-lógica usando funções temáticas

- `A Susana viu a chave`

```
(<PASS VER1> e1: [EXPERIMENTADOR (NOME s1 “Susana”)]
                 [TEMA <UM c7: {CHAVE1 CHAVE2 }>])
```

- `O coelho saltou para a horta. Estava faminto e as cenouras pareciam gordas e suculentas.`

```
(<PASS SALTAR1> e2: [AGENTE (O c8 COELHO1)]
                    [TO-LOC <O h1: HORTA1>])

(<PASS FAMINTO1> e3: [EXPERIMENTADOR c8])

(<PASS PGS1> e4: [EXPERIMENTADOR c8]
                 [TEMA <O cen1: CENOURAS1>]
                 [IN-LOC h1])
```

Problema 37 — Funções temáticas de sintagmas nominais

Os sintagmas nominais estão delimitados por parêntesis rectos:

- Vi [o homem]$_{\text{SN}}$ em [o monte]$_{\text{SN}}$ com [o telescópio]$_{\text{SN}}$

  [o homem] -> TEMA
  [o monte] -> EM-LOC
  [o telescópio] -> INSTRUMENTO

- [O dono]$_{\text{SN}}$ encontrou [o anel]$_{\text{SN}}$ debaixo de [o armário da cozinha]$_{\text{SN}}$

  [o dono] -> AGENTE
  [o anel] -> TEMA
  [o armário da cozinha] -> EM-LOC

- Durante [a noite]$_{\text{SN}}$, [o João]$_{\text{SN}}$ bebeu [o leite gelado]$_{\text{SN}}$ que estava em [o frigorífico]$_{\text{SN}}$

  [a noite] -> EM-TEMP
  [o João] -> AGENTE
  [o leite gelado] -> TEMA
  [o frigorífico] -> DE-LOC

Problema 38 — Funções temáticas associadas a verbos

Algumas funções temáticas que se podem associar aos verbos (as funções temáticas internas estão a negrito):

- Informar: **AGENTE**, **BENEFICIÁRIO**, TEMA, INSTRUMENTO

  [Eu]$_{\text{AGENTE}}$ informei [o dono]$_{\text{BENEFICIÁRIO}}$ que [não pagava a conta]$_{\text{TEMA}}$

  [A Joana]$_{\text{AGENTE}}$ informou-[me]$_{\text{BENEFICIÁRIO}}$ d[a sua decisão]$_{\text{TEMA}}$

  Ontem informei [o João]$_{\text{BENEFICIÁRIO}}$ d[o sucedido na escola]$_{\text{TEMA}}$

  [Ele]$_{\text{BENEFICIÁRIO}}$ já foi informado

  [Ela]$_{\text{BENEFICIÁRIO}}$ já [se]$_{\text{BENEFICIÁRIO}}$ informou

- Visitar: **AGENTE**, **TEMA/BENEFICIÁRIO**, EM-TEMP, EM-LOC, CO-AGENTE

  [O Zé]$_{\text{AGENTE}}$ visitou [o amigo]$_{\text{BENEFICIÁRIO}}$ n[o hospital]$_{\text{EM-LOC}}$ com a [Mãe]$_{\text{CO-GENTE}}$

  [A Leonor]$_{\text{AGENTE}}$ visita [os pais]$_{\text{BENEFICIÁRIO}}$ [todas as semanas]$_{\text{EM-TEMP}}$

  [Os franceses]$_{\text{AGENTE}}$ visitaram [o museu de arte antiga]$_{\text{TEMA}}$

  [Eu]$_{\text{AGENTE}}$ e [o João]$_{\text{CO-GENTE}}$ visitamos [o teatro]$_{\text{TEMA}}$

- Trocar: **AGENTE/EXPERIMENTADOR**, **TEMA**, CO-TEMA, CO-AGENTE, EM-LOC

  [O João]$_{\text{AGENTE}}$ trocou [o livro]$_{\text{TEMA}}$ por [um bilhete]$_{\text{CO-TEMA}}$

  [o Manuel]$_{\text{EXPERIMENTADOR}}$ trocou [o recado]$_{\text{TEMA}}$

  [A enfermeira]$_{\text{AGENTE}}$ troca de [turno]$_{\text{TEMA}}$ [todas as semanas]$_{\text{EM-TEMP}}$

  [A MARIA]$_{\text{AGENTE}}$ trocou [livros]$_{\text{TEMA}}$ com o [Zé]$_{\text{CO-AGENTE}}$

Problema 39 — Simplificação sintáctica de uma gramática com a característica SEM

Usando a convenção de:

- declarar separadamente as características cujo valor é idêntico na cabeça da regra e no constituinte principal
- Indicar o constituinte principal a itálico
- Omitir, sempre que possível o constituinte **VAR**, por este ser sempre obrigatório

obtém-se:

```
(F SEM (?sv ?sn)) -> (SN SEM ?sn) (SV SEM ?sv)

(SN VAR ?v2 SEM <?d ?v2 ?n>) -> (Det SEM ?d) (N SEM ?n)

(SN VAR ?v2 SEM (NOME ?v2 ?np)) -> (Det SEM ?d) (NP SEM ?np)

(SV VAR ?v1 SEM (λx.(?v ?v1 x))) -> (V SEM ?v SUBCAT _só)

(SV VAR ?v1 SEM (λx.(?v ?v1 x ?sn))) -> (V SEM ?v SUBCAT _sn) (SN SEM ?sn)

(SV VAR ?v1 SEM (λx.(& (?v ?v1 x ?sn) (?sp ?v1))) -> (V SEM ?v SUBCAT _snsp)
                                                      (SN SEM ?sn)
                                                      (SP SEM ?sp)

(SP VAR ?v2 SEM (λx.(?p x ?sn))) -> (P SEM ?p) (SN SEM ?sn)
```

"Características de cabeça" para F, SN, SV e SP: **VAR**

Usando agora a convenção de indicar entre parêntesis valores que só são usados por uma característica, omitindo a referência à característica, obtém-se:

```
(F SEM (?sv ?sn)) -> (SN SEM ?sn) (SV SEM ?sv)

(SN VAR ?v2 SEM <?d ?v2 ?n>) -> (Det SEM ?d) (N SEM ?n)

(SN VAR ?v2 SEM (NOME ?v2 ?np)) -> (Det SEM ?d) (NP SEM ?np)

(SV VAR ?v1 SEM (λx.(?v ?v1 x))) -> (V [_só] SEM ?v)

(SV VAR ?v1 SEM (λx.(?v ?v1 x ?sn))) -> (V [_sn] SEM ?v) (SN SEM ?sn)

(SV VAR ?v1 SEM (λx.(& (?v ?v1 x ?sn) (?sp ?v1))) -> (V [_snsp] SEM ?v)
                                                      (SN SEM ?sn)
                                                      (SP SEM ?sp)

(SP SEM (λx.(?p x ?sn))) -> (P SEM ?p) (SN SEM ?sn)
```

Problema 40 — Cálculo da semântica de várias frases

<table>
<tr><td colspan="3">CAT F
VAR v3
SEM (CAIR1 v3 (NOME v2 "Maria"))</td></tr>
<tr><td colspan="2">CAT SN
VAR v2
SEM (NOME v2 "Maria")</td><td>CAT SV
VAR v3
SEM (λx.(CAIR1 v3 x))</td></tr>
<tr><td>CAT Det
VAR v1
SEM O</td><td>CAT NP
VAR v2
SEM "Maria"</td><td>CAT V
VAR v3
SEM CAIR1
SUBCAT _só</td></tr>
</table>

1 a 2 Maria 3 cai 4

<table>
<tr><td colspan="5">CAT F
VAR v3
SEM ((PASS VER1) v3 <O v2 HOMEM1> (NOME v5 “João”))</td></tr>
<tr><td></td><td></td><td colspan="3">CAT SV
VAR v3
SEM (λx.((PASS VER1) v3 x (NOME v5 “João”)))</td></tr>
<tr><td colspan="2">CAT SN
VAR v2
SEM <O v2 HOMEM1></td><td></td><td colspan="2">CAT SN
VAR v5
SEM (NOME v5 “João”)</td></tr>
<tr><td>CAT Det
VAR v1
SEM O</td><td>CAT N
VAR v2
SEM HOMEM1</td><td>CAT V
VAR v3
SEM (PASS VER1)
SUBCAT _sn</td><td>CAT Det
VAR v4
SEM O</td><td>CAT NP
VAR v5
SEM “João”</td></tr>
</table>

1 o 2 homem 3 viu 4 o 5 João 6

<table>
<tr><td colspan="8">CAT F
VAR v3
SEM (& ((PASS DAR1) v3 <O v2 PROF1> <UM v5 RAPOSA1>)
(PARA-POSS1 v3 (NOME v8 “João”)))</td></tr>
<tr><td colspan="2"></td><td colspan="6">CAT SV
VAR v3
SEM (λx.(& ((PASS DAR1) v3 x <UM v5 RAPOSA1>)
(PARA-POSS1 v3 (NOME v8 “João”))))</td></tr>
<tr><td colspan="5"></td><td colspan="3">CAT SP
VAR v8
SEM (λx.(PARA-POSS1 x
(NOME v8 “João”)))</td></tr>
<tr><td colspan="2">CAT SN
VAR v2
SEM <O v2 PROF1></td><td></td><td colspan="2">CAT SN
VAR v5
SEM <UM v5 RAPOSA1></td><td></td><td colspan="2">CAT SN
VAR v8
SEM (NOME v8 “João”)</td></tr>
<tr><td>CAT Det
VAR v1
SEM O</td><td>CAT N
VAR v2
SEM PROF1</td><td>CAT V
VAR v3
SEM (PASS DAR1)
SUBCAT _snsp</td><td>CAT Det
VAR v4
SEM UM</td><td>CAT N
VAR v5
SEM RAPOSA1</td><td>CAT P
VAR v6
SEM PARA-POSS1</td><td>CAT Det
VAR v7
SEM O</td><td>CAT NP
VAR v8
SEM “João”</td></tr>
</table>

1 a 2 professora 3 deu 4 uma 5 raposa 6 a 7 o 8 João 9

Problema 41 — Rescrita de uma gramática usando as “convenções do Allen”

- (S INV - **VFORM** ?v{pres past} **AGR** ?a) -> (NP **AGR** ?a)
  (VP **VFORM** ?v{pres past} **AGR** ?a)
- (NP **AGR** ?a) -> (ART **AGR** ?a) (N **AGR** ?a)
- (NP **AGR** ?a) -> (PRO **AGR** M)
- (VP **AGR** ?a **VFORM** ?v) -> (V **SUBCAT** _none **AGR** ?a **VFORM** ?v)
- (VP **AGR** ?a **VFORM** ?v) -> (V **SUBCAT** _np **AGR** ?a **VFORM** ?v) NP
- (VP **AGR** ?a **VFORM** ?v) -> (V **SUBCAT** _vp:inf **AGR** ?a **VFORM** ?v) (VP **VFORM** inf)
- (VP **AGR** ?a **VFORM** ?v) -> (V **SUBCAT** _np_vp:inf **AGR** ?a **VFORM** ?v) NP
  (VP **VFORM** inf)
- (VP **AGR** ?a **VFORM** ?v) -> (V **SUBCAT** _adjp **AGR** ?a **VFORM** ?v) ADJP
- (VP **AGR** ?a **VFORM** inf **SUBCAT** inf) -> (TO **AGR** ?a **VFORM** inf) (VP **VFORM** base)
- ADJP -> ADJ
- ADJP -> (ADJ **SUBCAT** _vp:inf) (VP **VFORM** inf)